Mai/Juni 2017 | Nr. 555
www.elektormagazine.de

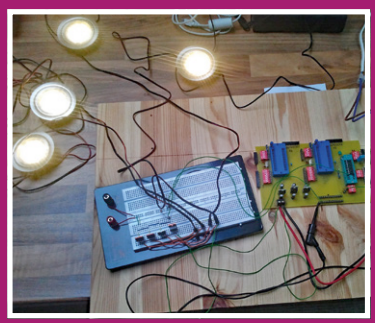
elektor

LEARN DESIGN SHARE

XBee/nRF24 Fernsteuerung

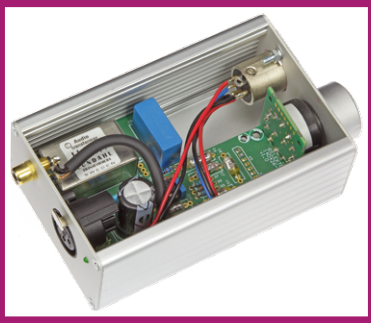
4 Kanäle und bidirektional

In dieser Ausgabe:
6 Labor-Projekte
9 Homelab-Projekte
3x Grundlagen
und vieles mehr!



Dimmbare Außenbeleuchtung
Für 12-V-LED-Lampen

Dev-Boards und Entwicklungskits
Für das Internet of Things



Mikrofon-Vorverstärker
Professionell, aber einfach

Li-Ion-Akkulader • Bluetooth-Modul fürs Steckbrett • **Grundlagen: Der I²C-Bus** • **Picowatt-Sender** • UV-Belichter mit PIC-Timer • **BBC micro:bit für Elektroniker: Einstieg mit mbed** • Automatische Rollladensteuerung • **NCSA II – verbesserter Network Connected Signal Analyzer** • Android auf dem Raspi: Display mit SPI ansteuern • **Grundlagen: LoRa** • **Treppenbeleuchtung mit LEDs** • Drehimpulsgeber anschließen ohne Mikrocontroller • **Tipps und Tricks: Löten mit Gullwing-Spitze** • **Bemerkenswerte Bauteile: Das Phantastron** • Retronik: 40 Jahre PET • **Web-Recherche: Experimentieren mit Tesla-Trafos**

G3078 E

(D) € 14,90 | CHF 27,00 | (A, B, L) € 16,00

4 198630 314904 06

INDUSTRIE 4.0 PRACTICE SUMMIT

AGILE STANDARDISIERUNG IM FOKUS DER PRAXIS

© shutterstock, Fraunhofer_IPA

19. – 20. Juni 2017
Vogel Convention Center
Würzburg

„Ein starker, innovativer Mittelstand, Spitzenplätze in Elektrotechnik, Automation und Industrie 4.0 werden unsere schwächere Position im „Internet of Everything“ auf Dauer ausgleichen können, wenn es uns gelingt neue Wertschöpfungsketten und Geschäftsmodelle zu etablieren.“

Dr.-Ing. Gunther Kegel
VDE Präsident, Vorsitzender der Geschäftsführung Pepperl+Fuchs, Mannheim

- Praxis und Orientierung
- Geschäftsmodelle
- Standardisierung
- Prüfung
- Smart Production
- Open Source, Plug Fest

ZIELGRUPPE

Experten und Entscheider aus den Bereichen: Produktion, Entwicklung und Konstruktion, Strategie, Innovations- und Technologiemanagement, Business Development, Forschung, IT, Produktmanagement, Key Account Management, Technischer Vertrieb

© VDE

www.vde.com/industrie4-0

#i40summit

PLATTFORM INDUSTRIE 4.0 · DIGITAL FACTORY JOURNAL · open automation.de · LNI4.0 LABS NETWORK INDUSTRIE 4.0 · STANDARDIZATION COUNCIL INDUSTRIE 4.0 · elektor · DVE · DIN · DKE · VDE

Impressum

48. Jahrgang, Nr. 555
Mai/Juni 2017

Erscheinungsweise: 12 x jährlich
(Elektor Doppelheft im Wechsel mit dem Elektor Business Magazin)

Verlag
Elektor-Verlag GmbH
Kackertstraße 10
52072 Aachen
Tel. 02 41/955 09 190
Fax 02 41/955 09 013

Technische Fragen bitten wir per E-Mail an redaktion@elektor.de zu richten.

Hauptsitz des Verlags
Elektor International Media
Allee 1, NL-6141 AV Limbricht

Anzeigen:
Margriet Debeij (verantwortlich)
Tel. 02 41/955 09 174 / Fax 02 41/955 09 013
Mobil: +49 (170) 550 539 6
E-Mail: margriet.debeij@eimworld.com

Julia Grotenrath
Tel. 02 41/955 09 177 / Fax 02 41/955 09 013
Mobil: +31 6 510 530 39
E-Mail: julia.grotenrath@eimworld.com

Es gilt die Anzeigenpreisliste ab 01.01.2017.

Distribution:
IPS Pressevertrieb GmbH
Postfach 12 11, 53334 Meckenheim
Tel. 0 22 25/88 01-0
Fax 0 22 25/88 01-199

Der Herausgeber ist nicht verpflichtet, unverlangt eingesandte Manuskripte oder Geräte zurückzusenden. Auch wird für diese Gegenstände keine Haftung übernommen. Nimmt der Herausgeber einen Beitrag zur Veröffentlichung an, so erwirbt er gleichzeitig das Nachdruckrecht für alle ausländischen Ausgaben inklusive Lizenzen. Die in dieser Zeitschrift veröffentlichten Beiträge, insbesondere alle Aufsätze und Artikel sowie alle Entwürfe, Pläne, Zeichnungen einschließlich Platinen sind urheberrechtlich geschützt. Ihre auch teilweise Vervielfältigung und Verbreitung ist grundsätzlich nur mit vorheriger schriftlicher Zustimmung des Herausgebers gestattet. Die veröffentlichten Schaltungen können unter Patent- oder Gebrauchsmusterschutz stehen. Herstellen, Feilhalten, Inverkehrbringen und gewerblicher Gebrauch der Beiträge sind nur mit Zustimmung des Verlages und ggf. des Schutzrechtsinhabers zulässig. Nur der private Gebrauch ist frei. Bei den benutzten Warenbezeichnungen kann es sich um geschützte Warenzeichen handeln, die nur mit Zustimmung ihrer Inhaber warenzeichengemäß benutzt werden dürfen. Die geltenden gesetzlichen Bestimmungen hinsichtlich Bau, Erwerb und Betrieb von Sende- und Empfangseinrichtungen und der elektrischen Sicherheit sind unbedingt zu beachten. Eine Haftung des Herausgebers für die Richtigkeit und Brauchbarkeit der veröffentlichten Schaltungen und sonstigen Anordnungen sowie für die Richtigkeit des technischen Inhalts der veröffentlichten Aufsätze und sonstigen Beiträge ist ausgeschlossen.

© 2017 elektor international media b.v.
Druck: Senefelder Misset, Doetinchem (NL)
ISSN 0932-5468

Zurück von der Embedded World

Auf der Messe in Nürnberg hatte ich Gelegenheit, mich in Sachen IoT nach Neuheiten umzuschauen. Mein Eindruck ist, dass die Vielfalt an Lösungen noch unüberschaubarer geworden ist. Beispiel ST: Der Halbleiterhersteller gab auf seinem Riesenstand wieder diversen Partnerfirmen ein Forum, und man konnte Stunden dort verbringen, um sich nur die dort gezeigten IoT-Plattformen vorführen zu lassen.
Unverkennbar ist der Trend zu immer ausgefeilteren Werkzeugen bei der Softwareentwicklung. Bevor man ein (IoT-)Projekt beginnt, setzt man diverse Checkboxen und umreißt damit, was der Prototyp können soll. Danach bindet das Entwicklungstool automatisch die nötigen Bibliotheken ein. Und das System weiß, welche Komponenten aufeinander aufbauen, macht selbst Vorschläge, fehlende Softwaremodule noch ins Projekt zu integrieren. So etwas konnte ich mir zum Beispiel am Stand des Betriebssystemherstellers Micrium ansehen. Dessen Konfigurationstool kann dabei sogar verschiedene Entwicklungsumgebungen der Controllerhersteller ansteuern und generiert dort automatisch ein Projektskelett, in das man „nur" noch den Applikationscode eingeben muss. Übrigens dürfen Hochschulangehörige und Maker das Betriebssystem nach Registrierung kostenlos in eigenen, nicht-kommerziellen Projekten verwenden (sicher einen Elektor-Artikel wert). Ohne automatische Projekt-Konfiguration, Code-Generierung und gekapselte Softwaremodule sind IoT-Projekte auf der Höhe der Zeit, die neben Protokollstapeln und Sicherheits-Features auch ein Interface zu einem Cloud-Speicher enthalten müssen, nicht mehr zu bändigen. Es mag für manchen Entwickler eine Umstellung bedeuten, dass er den Einblick in die Details verliert. Andererseits stört sich heute ja auch niemand mehr daran, komplexe ICs einzusetzen – Hardwaremodule, die vielerlei Funktionen kapseln und von denen oft nur noch die Schnittstellen zur Außenwelt bekannt sind.
Spannende Zeiten – bleiben Sie dran!

Jens Nickel
Chefredakteur Elektor

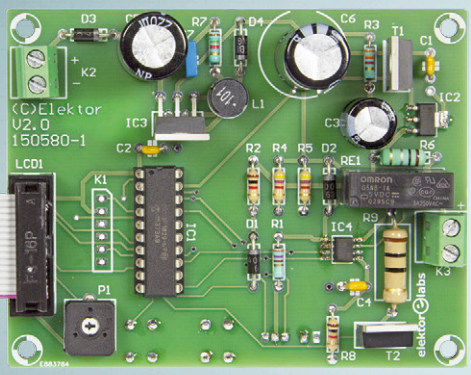
Unser Team

Chefredakteur:	Jens Nickel (v.i.S.d.P.) (redaktion@elektor.de)
Ständige Mitarbeiter:	Dr. Thomas Scherer, Rolf Gerstendorf
Leserservice:	Ralf Schmiedel
Korrekturen:	Malte Fischer
Internationale Redaktion:	Thijs Beckers, Jan Buiting, Mariline Thiebaut-Brodier
Elektor-Labor:	Ton Giesberts, Luc Lemmens, Denis Meyer, Jan Visser, Clemens Valens
Grafik & Layout:	Giel Dols

Inhalt

48. Jahrgang – Nr. 555
Mai/Juni 2017

118



Rubriken

3 Impressum

26 Tipps und Tricks
Von Lesern für Leser

63 HomeLab Helikopter

66 Bemerkenswerte Bauteile
Das Phantastron

88 Projekt 2.0
Korrekturen, Updates und Leserbriefe

90 Retronik
Alles Gute zum 40., PET!

94 Q&A
Smarte Gebäude

99 Elektormagazine.de/Labs
Projekte in der Pipeline

100 Web-Recherche
Experimentieren mit Tesla-Trafos

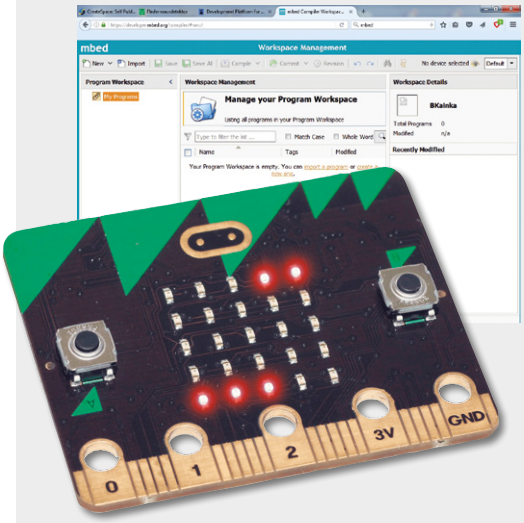
126 Elektor Ethics
Brauchen wir ein Robotergesetz?

130 Hexadoku

BBC micro:bit für Elektroniker

20

Einstieg mit Mbed



Der BBC micro:bit ist nicht nur für Schüler geeignet, er ist auch ein tolles Controllerboard für Elektroniker. Auf kleinstem Raum ist fast alles vorhanden, was man immer wieder braucht: Digitale Ein-/Ausgänge mit PWM, AD-Wandler sowie eine USB-Schnittstelle für Stromversorgung, Programmierung und Datenaustausch. Verschiedene Sensoren und Bluetooth runden die Ausstattung ab.

Hintergrund

6 Sprungbrett ins IoT
Dev-Boards und Entwicklungskits für das Internet of Things

40 USB-Mikroskop Andonstar V160

42 Der I²C-Bus
Teil 1: Protokoll

82 Android auf dem Raspi (2)
Display mit SPI ansteuern

105 Grundlagen: LoRa
Niedrige Geschwindigkeit, hohe Reichweite

116 Aus dem Labor
Arbeiten mit dem eC-stencil-fix

Projekte

14 PSoC-BLE-Modul im L-Board-Format
Ideal fürs Prototyping – passend fürs Steckbrett

20 BBC micro:bit für Elektroniker
Einstieg mit mbed

28 Mikrofon-Vorverstärker
Professionell, aber einfach

6

Dev-Boards und Entwicklungskits für das Internet of Things

Li-Ion-Akkulader

Ladegerät mit korrekten Kennlinien

- **Lädt beliebige Li-Ion-Akkus mit einer Nennspannung von 3,6 V**
- **Formatierungsmodus für Rettungsversuche tiefentladener Zellen**
- **Entlademodus zur Ermittlung der realen Akkukapazität**
- **Alle wichtigen Informationen verfügbar dank LC-Display**

Mikrofon-Vorverstärker

28

Professionell, aber einfach

Mit einem Kassettenrekorder nimmt niemand mehr auf – heute kommt oft ein Notebook zum Einsatz. Doch während die Line-Eingänge vieler PCs zumeist eine gute Klangqualität aufweisen, befinden sich die Mikrofon-Eingänge vieler Notebooks nur auf einem mittelmäßigen Niveau, das durch Rauschen und Verzerrungen gekennzeichnet ist. Außerdem lässt sich in der Regel nur ein Mikrofon anschließen - von professionellen Mikrofonen mit symmetrischen Anschlüssen ganz zu schweigen, die außerdem oft noch eine Phantomspeisung von 48 V benötigen. Hier kommt ein Vorverstärker, der diese Nachteile nicht aufweist und vor allem die Besitzer von High-End-Soundkarten begeistern wird.

46 NCSA II – verbesserter Network Connected Signal Analyzer
Schneller, neue Fähigkeiten und einfacherer Einsatz

54 Vier-Kanal-Fernsteuerung
In zwei Varianten: XBee oder nRF24

68 IoT-Gateway und drahtlose Endknoten
Teil 2: Die Software

74 Dimmbare Außenbeleuchtung
Einstellbare Helligkeiten für 12-V-LED-Lampen

79 Anwesenheitssimulator
Vorbeugen gegen Einbruch

96 WaterDeck
Analyse des Wasserverbrauchs über das Internet

102 Treppenbeleuchtung mit LEDs
Ein Controller für viele Leuchtdioden

110 Picowatt-Sender
UKW-Sender mit BF961

112 UV-Belichter mit PIC-Timer
Matrix aus 54 UV-LEDs

114 Drehimpulsgeber anschließen…
…ohne Mikrocontroller!

118 Li-Ion-Akkulader
Ladegerät mit korrekten Kennlinien

Vorschau

Das nächste Heft ist wie immer randvoll gefüllt mit Schaltungsprojekten, Grundlagen sowie Tipps und Tricks für Elektroniker.

Aus dem Inhalt:

- Kurzwellen-Transceiver mit FPGA
- High-End-Audio-DAC für Raspberry Pi
- Entwicklungssystem für Smartwatches
- Ultraschall-Generator
- 7-Segment-Anzeige für Torstand und Co.
- Tipps für das Platinendesign
- Kennlinienschreiber für Röhren
- iPhone-Programmierung für Elektroniker

Und vieles mehr!

Änderungen vorbehalten.
Elektor Juli/August 2017 erscheint am 21. Juni 2017.
Verkaufsstellen findet man unter www.pressekaufen.de.

Vorschau Elektor Business

Sensoren & Messtechnik

- Stromversorgung von Sensoren
- Hochgenaue LCR-Meter
- Beschleunigungsmesser
- Überwachung von Elektromotoren

Und vieles mehr!

Änderungen vorbehalten.
Elektor Business 3/2017 erscheint am 17. Mai 2017.

Sprungbrett ins IoT

Dev-Boards und Entwicklungskits für das Internet of Things

Von **Viacheslav Gromov** (D), readers@gromov.de

Das Internet of Things (und hier vor allem die intelligente Haus-Automatisierung) gehört zu den beliebtesten Elektronik-Themen bei unseren Lesern. Am Anfang musste man Systeme für die Entwicklung und das schnelle Prototyping noch selbst zusammenstellen. Heute überzeugt die ganze Embedded-Branche mit unglaublich kostengünstigen Entwicklungskits, die alles enthalten, was das Entwicklerherz begehrt. Innerhalb von wenigen Minuten können selbst Einsteiger ihren ersten Prototyp entwickeln.

IoT-Entwicklungskits enthalten heutzutage meist alles, was man zum Entwickeln benötigt – vom USB-Kabel über das Board mit einer Menge Sensoren bis hin zum nötigen Akku und Gehäuse. Doch genau wegen dieser enormen Vielfalt verliert man sich schnell in der unübersichtlichen Welt des Internets der Dinge: Welchen Prozessor verwende ich? Welche Sensoren? Welche Funktechnologien? Welche Art der Stromversorgung? Und dazu kommen dann die üblichen Optimierungsgrößen wie Kosten, Entwicklungszeit, Stromverbrauch und vieles mehr. Der Markt ist bis zum Rande gefüllt – um nicht von „gesättigt" zu reden – mit Plattformen für alle gängigen Technologien und Anwendungsfelder.
Die bekannten, mehr oder weniger professionellen IoT-Plattformen wie Micro:Bit, einige Arduino-Typen, die zahlreichen ESP8266-Boards und viele mehr sollen in dieser (bei weitem nicht vollständigen) Übersicht ausgeklammert werden, statt dessen konzentriert sie sich eher auf die Lösungen der großen Halbleiterhersteller. Schließlich liegt hier der Ursprung der sich rasch weiterentwickelnden IoT-Technologien. Vom Prozessor über Sensoren bis hin zu Wireless-Bausteinen, brandneu oder nur aktuell, gängig oder relativ unbekannt, günstig oder teuer, universell oder auf bestimmte Anwendungsgebiete ausgerichtet:

Die Übersicht zeigt auf den folgenden Seiten stichprobenartig gemischt die für das IoT wirklich relevanten Stärken der Produkte und die Eigenschaften der Hard- und Software. Bestimmt ist auch etwas für Sie dabei!

Hexiwear von MikroElektronika und NXP

NXP (früher Freescale) hat zwar auch eigene Entwicklungsboards für das Internet of Things, noch interessanter sind aber die Kooperationen mit anderen Herstellern oder Distributoren. So ist Hexiwear entstanden (**Bild 1**) [1]. Die Plattform wird zwar von MikroElektronika produziert und vermarktet, von NXP stammen aber die MCUs und die meisten Sensoren. Darüber hinaus unterstützt NXP diese Plattform auch sehr intensiv, etwa durch die Anpassung des Kinetis-Design-Studios. Aber was ist nun Hexiwear? Es ist ein größeres Gadget in Form einer Armbanduhr, das in verschiedene Gehäuse und Armbänder verbaut (Wearable), aber auch stand alone mit vielen möglichen Erweiterungen betrieben werden kann. Somit ist das Hexiwear-Board sehr vielfältig, auch weil es über BLE, USB, viele Sensoren (Puls, Licht, Temperatur, Luftdruck, Luftfeuchtigkeit, Beschleunigung, Gyroskop und Magnetfeld), ein 1,1-Zoll-OLED-Farbdisplay und einen Akku, dessen 190 mAh bei regelmäßiger Nutzung mehrere Stunden ausreichen, verfügt. Die Steuerung erfolgt durch die sechs Touch-Bedienelemente auf der Front, bei Bedarf kann aber auch die Bedienung mit dem eingebauten Feedback-Vibrationsmotor intensiviert werden. Auf der Front befindet sich darüber hinaus eine programmierbare RGB-LED.
Im Inneren werkeln zwei MCUs: der gut bekannte K64F (M4F, 120 MHz, 1 MB Flash), der als Hauptcontroller die Zentrale des Boards darstellt und alle angeschlossenen Peripherieelemente ansteuert sowie der KW40Z (M0+, 48 MHz, 160 KB) aus der Dual-Mode-Funkserie (BLE + IEEE 802.15.4), der bei Hexiwear für die BLE-Verbindung (**B**luetooth **L**ow **E**nergy) sorgt. Das Board besitzt eine 50-polige Anschlussleiste auf der Unterseite mit den wichtigsten (Schnittstellen-)Anschlüssen, vor allem für das zusätzlich erhältliche Docking-Station-Board, auf dem drei User-Taster und -LEDs zu finden sind, aber auch je ein Reset-Taster für die beiden MCUs, ein MicroSD-Kartenslot, ein I²S-, ein MicroUSB- und ein SWD-Anschluss. Im Mittelpunkt dieses Erweiterungsboards stehen aber die drei Slots, auf die die populären MikroElektronika-ClickBoards gesteckt werden können. Es gibt hunderte verschiedener ClickBoards mit den verschiedensten Sensoren, Aktoren und Schnittstellen. Man gelangt also mit Hexiwear und den nötigen ClickBoards schnell und günstig zu eigenen Prototypen.
Ab Werk läuft auf dem Hexiwear eine Firmware, die alle (Sensor-) Funktionen demonstriert. Außerdem kann man das Hexiwear per Bluetooth mit der Hexiwear-App von WolkAbout auf einem Smartphone verbinden, wobei man die Daten live verfolgen, in die WolkAbout-Cloud hochladen und anschließend in verschiedenen Ansichten visualisieren kann. Ein Firmware-Update kann auch drahtlos durchgeführt werden (OTAP = Over-The-Air-Programming). Programmiert wird das Hexiwear am besten über seine Micro-USB-Schnittstelle, wobei ein OpenSDA-Debugger die Brücke zu den MCUs darstellt. Es gibt drei Varianten [2], eigene Software für Hexiwear zu schreiben: Für den Einstieg

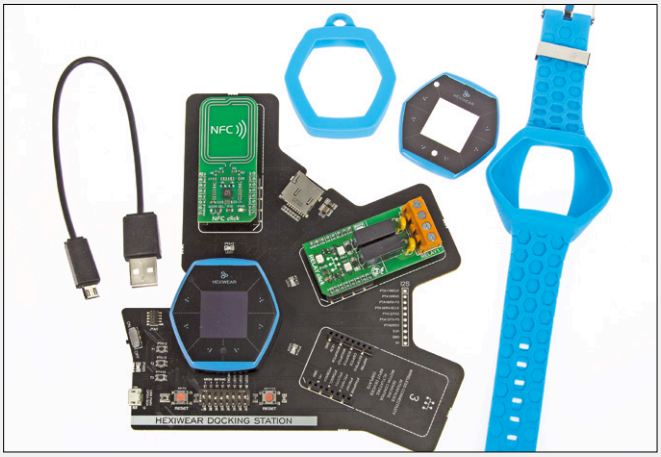


Bild 1. Hier ist das Hexiwear auf seiner Docking-Station mit zwei möglichen Click-Erweiterungsboards zu sehen. Daneben liegt Zubehör wie das USB-Kabel oder das zusätzliche Color-Pack.

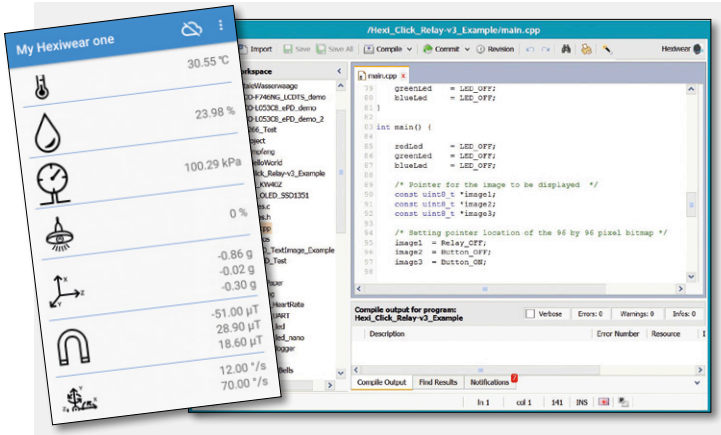


Bild 2. Der Screenshot links zeigt die Hexiwear-App mit aktuellen Daten, rechts ein Codeausschnitt aus der mbed-Oberfläche, in dem die einfachen und strukturierten Befehle gut zu erkennen sind.

kann man die einfache mbed-Onlineumgebung (**Bild 2**) nutzen, für Linux-Liebhaber gibt es die Distribution Zephyr und wer die Hardwaremöglichkeiten ausreizen möchte, kann zum **K**inetis **D**esign **S**tudio (KDS) greifen. Auf Hackster.io findet man viele IoT-Hexiwear-Projekte zur Inspiration. Die Dokumentation und die Versorgung mit Software ist sehr löblich (alles open-source!). Der Preis des puren Hexiwear liegt bei 50 €, mit Docking Station und dem Gehäuse-Zubehör kostet es ungefähr das Doppelte.

Wenn man schon über NXP redet, darf man das kleine WaRP7-Board (**Bild 3**) nicht vergessen, das in Kooperation mit Farnell/Element 14 [3] entstanden ist. Geliefert wird es mit einem 180-mAh-Akku. Es verfügt über USB, NFC, BLE und WLAN als drahtlose Schnittstellen. Außerdem sind eine Kamera und viele Sensoren mit an Bord. Der Prozessor i.MX7 Solo (A7 + M4) wurde um einige Speicher-ICs erweitert. Es gibt ebenfalls einen Slot für ClickBoards. Als Zusatz kann noch ein rundes Farbtouchdisplay angeschlossen werden. Diese Plattform ist empfehlenswert für Entwickler, die gerne über die Befehlszeile und mit Linux-Distributionen arbeiten und dabei über ein fundiertes Grundwissen verfügen. Alternativ wäre auch Android als Betriebssystem denkbar. Der Preis beläuft sich auf ungefähr 90 €.

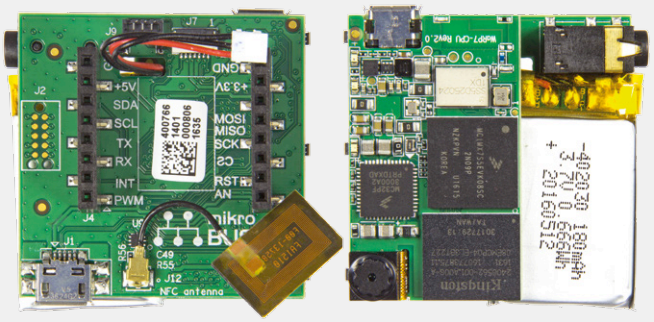


Bild 3. Hier ist das WaRP7-Board von beiden Seiten zu sehen.

SensorTile von STMicroelectronics

STMicroelectronics (ST) hat neuerdings ein sehr interessantes Kit namens SensorTile [4] auf den Markt gebracht, dessen Preis bei etwa 80 € liegt (**Bild 4**). Im Fokus steht das münzgroße Sensortile, das auf einem STM32L476-Mikrocontroller (M4F, 80 MHz, 1 MB Flash) basiert und einen BlueNRG-MS für die BLE-Kommunikation enthält. Es verfügt über ein Mikrofon, ein Barometer und einen Beschleunigungssensor, ein Gyroskop

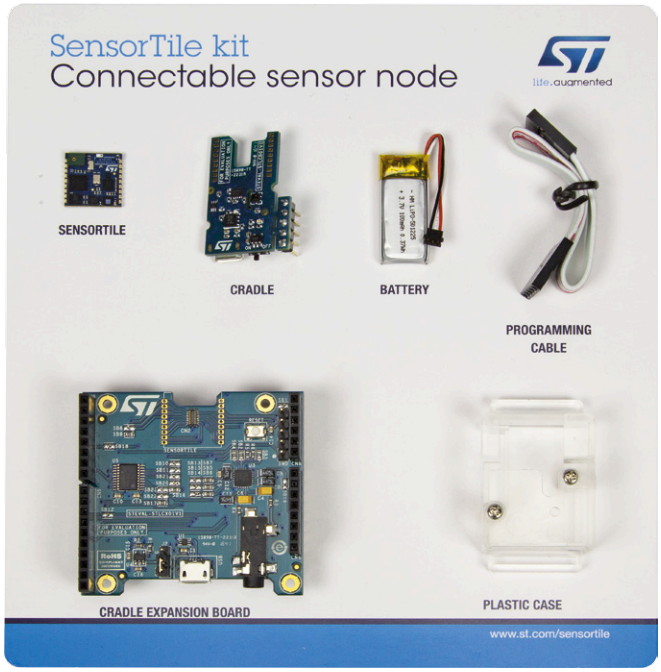


Bild 4. Das SensorTile-Kit mit all seinen Bestandteilen.

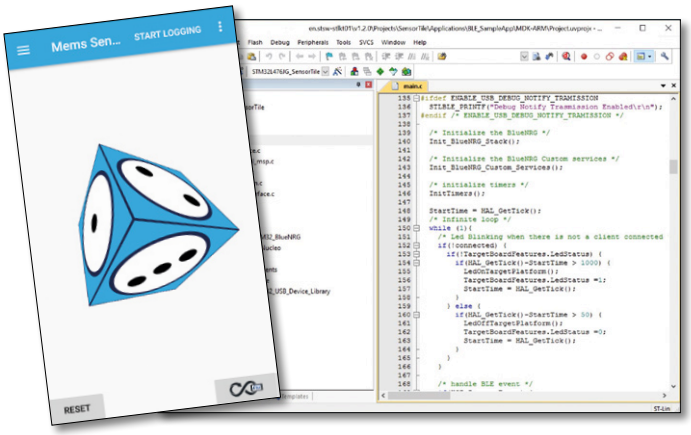


Bild 5. Links ein Beispiel für den Beschleunigungssensor in der Smartphone-App, rechts ein Ausschnitt aus der Starter-Beispielvorlage in der Keil-Umgebung.

und einen Magnetfeldsensor. Die wichtigsten Anschlüsse sind auf die Lötpads an den Seiten herausgeführt, aber auch auf den kleinen Anschluss auf der Unterseite des Boards.

Da das Sensortile ohne Stromversorgung nicht arbeiten kann, sind zwei Adapterboards im Kit enthalten. Das eine heißt Cradle, das man an das Sensortile-Board löten muss. Cradle erweitert das Sensortile um einen Feuchtigkeits- und Temperatursensor, einen MicroSD-Kartenslot, einen SWD-Programmieranschluss sowie einen Micro-USB-Ladeanschluss. Im Lieferumfang steckt auch ein 100 mAh großer Akku, der, auf das Cradle gesteckt, das Sensortile auf dem Cradle mit Strom versorgt. Der ganze Aufbau kann im ebenfalls mitgelieferten Klarsichtgehäuse untergebracht werden.

Die zweite Adapter-Lösung ist ein Arduino-Erweiterungsshield mit Audio-DAC (Klinkenausgang) und ebenfalls mit einem Micro-

USB- und SWD-Anschluss. Dieses Shield kann sowohl mit dem Sensortile im Stand-alone-Modus betrieben als auch auf ein Nucleo- oder ein anderes Board im Arduino-Design aufgesteckt werden, sodass das Sensortile von einer anderen MCU gesteuert werden kann (und umgekehrt).
Beide Adapterboards verfügen über den SWD-Anschluss, über den das Sensortile programmiert werden muss, am besten mit einem ST-Link, der ja auch auf jedem 64-Pin-Nucleo-Board zu finden ist. Dem Kit liegt dazu ein Brückenkabel bei. Auch ST hat eine App namens ST BlueMS entwickelt, die „out of the box" die Fähigkeiten des Sensortiles demonstriert (**Bild 5**). Für viele Sensor-Beispiele werden zuerst aber noch Lizenzen angefragt, was aber automatisch per E-Mail geschieht und nicht viel Zeit in Anspruch nimmt. In der App ist auch ein Daten-Upload in die Watson-IoT-Cloud von IBM möglich.
Die Software kann man mit den üblichen STM32-fähigen Compilern erstellen. Dabei lässt man sich von der etwas komplexeren BlueMicrosystem2-Vorlage leiten, die genau das Beispielprogramm, das ab Werk auf dem Sensortile läuft, darstellt. Oder man verwendet die auf die Grundfunktionen vereinfachte Vorlage „Starter Firmware", für die es auch zum Einstieg drei Beispielprogramme gibt.

Solar Powered IoT Device Kit
von Cypress Semiconductor

Bestünde dieses Kit [7] für rund 50 € nur aus dem BLE-Modul von Cypress, würde es gar nicht aus der Flut der IoT-Lösungen herausragen. Das Cypress BLE-Modul ist zwar gut und die Entwicklungsumgebung PSoC-Creator ist für solche Anwendungen sehr gut geeignet, der Knüller ist aber die Solarzelle mit dem Energy-Harvesting-Design auf dem Board! Denn die Trends beeinflussen sich gegenseitig: Das Internet der Dinge befeuert die Power-Optimierung und diese mündet manchmal in Energy-Harvesting, wenn es möglich und günstig ist. Energy-Harvesting ist im IoT nicht zu vernachlässigen, sondern steht im Gegenteil hoch im Kurs.
Dieses Kit zeigt das Können des EZ-BLE PRoC Moduls (CYBLE-022001-00, M0, 48 MHz, 256 KB Flash). Zusammen mit diesem Modul befinden sich nur noch die Energy-Harvesting-Schaltung und ein paar Sensoren für Temperatur, Feuchtigkeit und Licht auf der Platine, aber weitere Sensoren können angeschlossen werden. Mitgeliefert wird noch eine USB-BLE-Platine, die dann über den Computer mit der Energy-Harvesting-Platine kommunizieren kann (**Bild 8**). So lässt sich neben dem üblichen Datenfluss auch die Reichweite und Signalstärke bestimmen, die bei bestimmten Harvesting-Varianten (außer Solar zum Beispiel auch Piezo) in verschiedenen Umgebungen maximal erreicht werden kann. Der PRoC an der Harvesting-Platine kann durch einen (nicht mitgelieferten) externen Debugger (am besten MiniProg 3 am vorgesehenen SWD-Anschluss) und dem modular und ziemlich einfach aufgebauten PSoC-Creator [8] programmiert werden (**Bild 9**). Im Lieferumfang enthalten sind noch ein paar bedrahtete Bauteile, die für den Einstieg mit den Beispielprojekten wichtig sind.

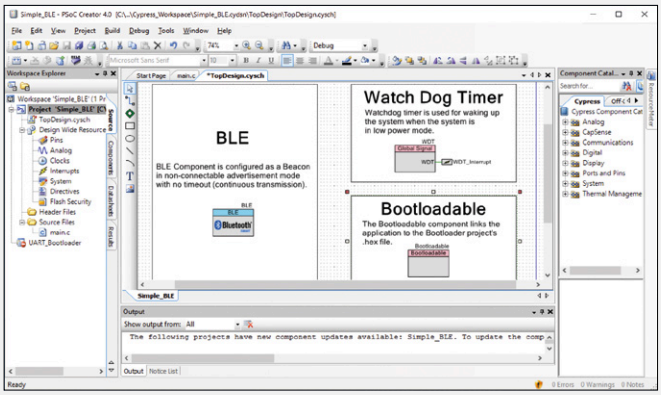


Bild 9. Dank des „LEGO-Prinzips" des PSoC-Creators fällt auch die BLE-Programmierung leicht – alles ist schon automatisch vorkonfiguriert.

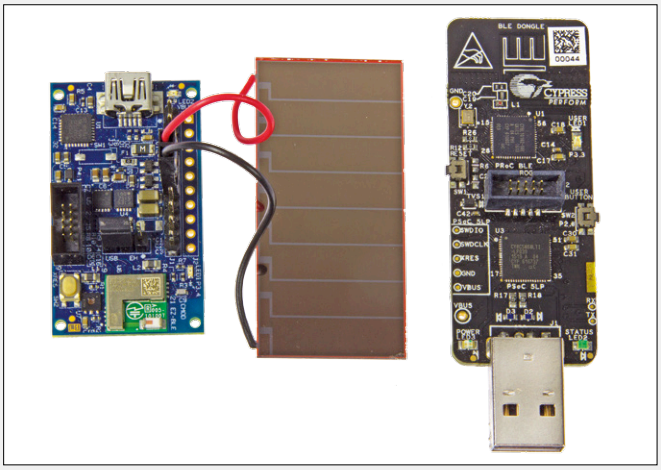


Bild 8. Hier sind die beiden beschriebenen Platinen sowie die angeschlossene Solarzelle des Device Kits zu erkennen

SimpleLink SensorTag von Texas Instruments

Schon für rund 30 € können Sie einen der SensorTags von TI (**Bild 10**) bekommen [9]. Wenn Sie es auspacken, kommt ein streichholzschachtelgroßer Anhänger mit Gummigehäuse zum Vorschein. Dies klingt zunächst harmlos und langweilig, beim zweiten Blick fällt jedoch auf, was der kleine Anhänger so alles drauf hat: Nach der Installation der App namens „SensorTag" von TI (**Bild 11**) wird es deutlich: Man kann alle Tag-Sensor-Daten live per BLE oder WLAN (je nach Tag) abrufen, sie in die IBM-Watson-IoT-Cloud hochladen und dort auswerten.
Eines der neuesten Tags wird von einer CC1350-Dual-Mode-MCU (M3, 48 MHz, 128 KB) gesteuert, die BLE unterstützt und zusätzlich auf dem 1-GHz-Sub-Band auf langen Strecken bis 2 km (!) kommunizieren kann. Von außen sieht man nur einige Taster, aber hinter den schmalen, durchsichtigen Öffnungen des inneren Plastikgehäuses verbirgt sich jede Menge Sensorik: ein Mikrofon, ein Licht-, ein Feuchtigkeits-, ein Luftdruck-, und ein Temperatursensor (normal und infrarot) und schließlich ein Beschleunigungs-, ein Gyroskop- und ein Magnetfeldsensor. Eine User-LED gibt es ebenfalls. Die kleine Knopfzellenbatterie

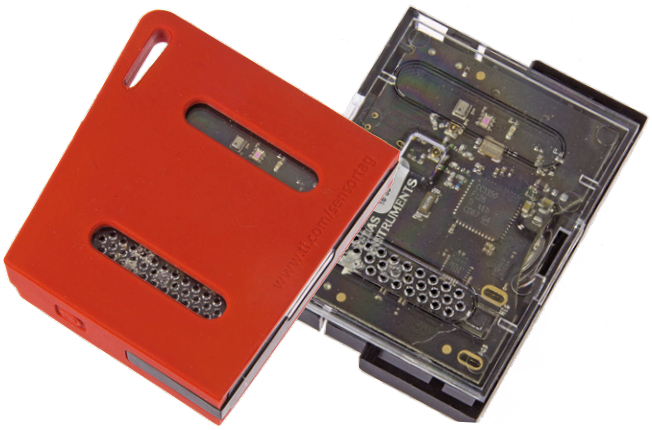


Bild 10. Hinter dem Gehäuse verbirgt sich das CC1350-SensorTag.

auf der Rückseite kann aufgrund des Low-Power-Konzepts bei nicht so häufigem Gebrauch ein Jahr halten.
Dies klingt schon so, als wäre der Anhänger ab Werk schon fertig und unveränderbar, aber dies ist natürlich nicht der Fall. Über den herausgeführten JTAG-Anschluss kann man diesen Anhänger mit dem **C**ode-**C**omposer-**S**tudio (CCS, Wireless-kompatible Ausführung) von TI [10] programmieren. Da die Hardware schon fertig ist, braucht man sich in der Prototyp-Phase nur um die Software zu kümmern. Dazu stehen viele Bibliotheken und eine ausführliche Dokumentation bereit, selbst für die App-Programmierung für das Smartphone.
Es gibt noch eine Reihe an zusätzlichen Softwaretools wie das Sensor-Controller-Studio, das Sensor-Anwendungen für Mikrocontroller (vor-)konfiguriert. Die SensorTags unterstützen übrigens auch eine drahtlose Programmierung (OTAP): In der App kann man beispielsweise per BLE die Tag-Firmware updaten. Zusätzlich besteht seit neuestem auch die Möglichkeit, am 20-poligen Tag-Anschluss Erweiterungsplatinen, die sogenannten DevPacks anzuschließen, zum Beispiel ein Display oder eine RGB-Leuchte. Das Angebot an diesen DevPacks wächst stetig. Im Gummi-Gehäuse wird es dann allerdings recht eng, so dass man es an bestimmten Stellen mit einem scharfen Messer und ruhiger Hand zurechtschneiden muss – das bisschen Handarbeit beherrscht aber jeder.
Besonders hervorzuheben ist das etwa 15 € teure Debugger-DevPack, mit dem man das Tag per USB-Schnittstelle bequem programmieren kann. Große oder teure Debugger sind damit völlig überflüssig. Insgesamt ist das Preis/Leistungsverhältnis hervorragend. Neben dem CC1350 gibt es noch den CC2650 mit BLE und ZigBee oder 6LoWPAN (das schon erwähnte IEEE 802.15.4); ein WLAN-Tag soll laut Hersteller in Kürze folgen.

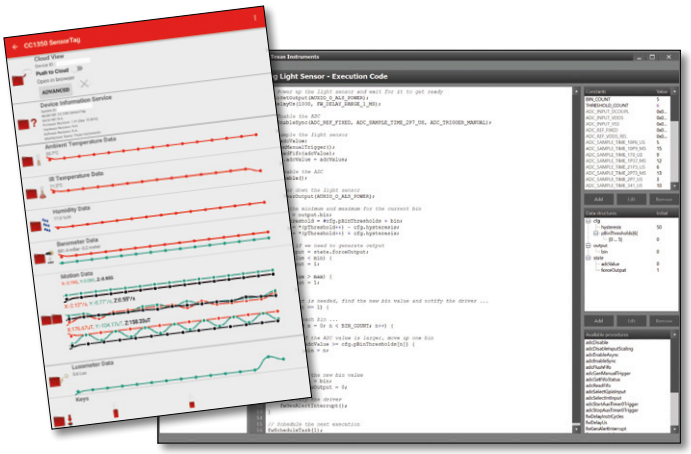


Bild 11. Hier ist links die SensorTag-App-Oberfläche zu sehen, rechts ein Screenshot aus dem Sensor-Controller-Studio.

XDK von Bosch Connected Devices and Solutions

Das XDK110 (**Bild 6**) von Bosch gibt es schon länger, nichtsdestotrotz steht es immer noch hoch im Kurs [5]. Die hohe Qualität von Gehäuse, Zubehör, Onlineauftritt und der Platine

Bild 6. Die wichtigsten Bestandteile des XDK110-Kits; sogar eine (Wand-)Halterung ist vorhanden. Dieses Bild zeigt, wie erstaunlich klein dieser Winzling wirklich ist.

selbst bezahlt man mit knapp 200 €. Bosch ist weltweit anerkannt für seine vorzüglichen Sensoren, von denen eine große Anzahl im XDK verbaut ist. Das XDK selbst steckt in einem Plastikgehäuse mit vier LEDs, zwei Tastern und einer durchsichtigen Öffnung für die Sensoren (Magnetfeld, Beschleunigung, Inertial, Gyroskop, Licht, Temperatur, Luftfeuchtigkeit, Luftdruck und Audio). An der Seite befinden sich ein MicroSD-Kartenslot, ein Programmieranschluss sowie eine 26-polige Anschlusswanne für das Verbindungskabel zum T-Board (beides wird mitgeliefert). Dieses T-Board („XDK Gateway“) kann zum Beispiel ganz einfach auf eine Steckplatine gesteckt werden, um erste Prototypen aufzubauen. Der XDK übernimmt also auch komplexere Kontroll- und Steueraufgaben außerhalb seines Gehäuses, denn der verbaute ARM-Cortex-M3-Mikrocontroller mit 1 MB Flash-Speicher besitzt eine ausreichend hohe Rechenleistung. Sollte der Datenspeicher nicht ausreichen, kann er vollständig auf eine optionale MicroSD-Karte ausgelagert werden. Der Akku mit 560 mAh macht mobile Anwendungen über längere Zeitspannen möglich. Programmiert wird dieser Baustein mit dem kostenlosen XDK-Workbench [6]. Dort findet man viele Beispielprogramme und viele dazugehörige Schritt-für-Schritt-Anleitungen. Das Board verfügt über BLE und WLAN, wodurch sowohl die Verbindung zum Smartphone (siehe VirtualXDK-Beispielprogramm mit App in **Bild 7**) und zu anderen IoT-Devices möglich ist. Gleichzeitig kann so auch auf

das Internet zugegriffen werden. Auch andere Softwarehersteller wie Relayr unterstützen das XDK mit ihren eigenen Online-Lösungen (Cloud, Analyse und so weiter). Zur Programmierung wird normalerweise der ab Werk vorprogrammierte Bootloader und die MicroUSB-Schnittstelle verwendet, nur in Ausnahmefällen muss man zum (externen) Debugger für die vorgesehenen Programmieranschlüsse greifen.

Abschließend kann man sagen: Wenn man nach einer professionellen, gut durchdachten und unterstützten (Industrie-)Plattform sucht, wird man trotz des relativ hohen Preises vom XDK nicht enttäuscht – eher das Gegenteil ist der Fall! ◀

(160343)

Bild 7. Links: Die Hauptansicht der VirtualXDK-App. Die Eclipse-basierte XDK-Workbench auf der rechten Seite sieht für den erfahrenen Entwickler sehr vertraut aus; nur die Programmierung beziehungsweise der Umgang mit dem Bootloader muss erlernt werden. Dies sollte aber keine unüberwindliche Hürde darstellen.

Was gibt es noch?	
AT88CKECC-AWS-XSTK von Microchip, http://goo.gl/FPXVoi	Dieses gut 200 € teure modulare Kit zeigt eine der neuesten IoT-Sicherheitstechnologien von Microchip und die Zusammenarbeit mit den **A**mazon **W**eb **S**ervices (AWS).
P-NUCLEO-LRWAN1 von ST, http://goo.gl/VLyLCT	Im rund 40 € teuren Kit ist ein L073RZ-Nucleo-Board mit einem LoRa-Expansion-Shield enthalten, was den neuesten Trend auf dem IoT-Markt leicht zugänglich macht.
P-NUCLEO-USB001 von ST, http://goo.gl/wjDplk	USB-C ist eine typische Schnittstelle (auch zum Laden!) bei modernen IoT-Devices, das F072RB-Nucleo-Board mit einem Power-Delivery-Expansion-Board für circa 50 € demonstriert diese Möglichkeiten.
C027 mbed-basiertes IoT-Kit von u-blox, http://goo.gl/QSz6Oh	Dieses (je nach Ausführung) um 100 € teure IoT-Board legt den Schwerpunkt auf GPS-Ortung beziehungsweise UMTS/GSM-Empfang, was bei Smart-Devices oft notwendig ist.
Thunderboard React/Sense Kit von Silicon Labs, http://goo.gl/dfGhfy	Diese beiden kleinen Boards im 30-Euro-Bereich mit zahlreichen Sensoren und anderer Peripherie wenden je nach Typ unterschiedliche Funktechnologien an und haben ein gutes Preis-Leistungs-Verhältnis.
NuMaker Uni von Nuvoton, http://goo.gl/Nm9XkG	Im untersten Preissegment gibt es von diesem chinesischen Hersteller einige IoT-Boards aus der NuMaker-Serie, darunter auch dieses mit Sensoren, Bluetooth und WLAN für unter 30 €.
Synergy S3A7 IoT Fast Prototyping Kit von Renesas, http://goo.gl/ov4UsG	Dieses modulare IoT-Kit für gut 150 € mit einem relativ großen Display kann aufgrund des (Sandbox-)Konzepts um viele verschiedene Sensoren und Funkmodule erweitert werden.
…und vieles mehr!	

Weblinks

[1] www.hexiwear.com

[2] www.hexiwear.com/getting-started/

[3] www.element14.com/community/docs/DOC-79058/l/warp7-the-next-generation-iot-and-wearable-development-platform

[4] www.st.com/sensortile

[5] http://xdk.bosch-connectivity.com/

[6] http://xdk.bosch-connectivity.com/software-downloads

[7] www.cypress.com/documentation/development-kitsboards/s6sae101a00sa1002-solar-powered-iot-device-kit

[8] www.cypress.com/products/psoc-creator-integrated-design-environment-ide

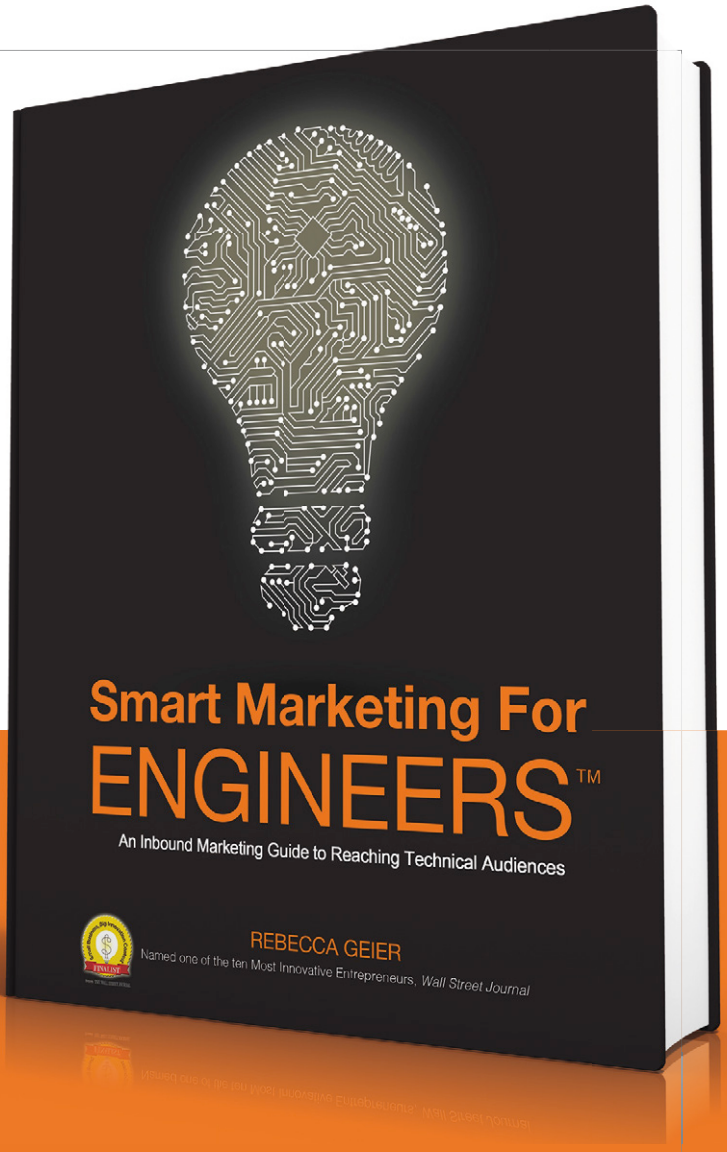
[9] www.ti.com/ww/en/wireless_connectivity/sensortag2015/

[10] www.ti.com/tool/ccstudio-wcs

Elektor präsentiert: **Rebecca Geiers**

Workshop Tour
Smart Marketing für Ingenieure

Mit der Verbreiterung von Marketing-Kanälen und der Hinwendung zum Online-Kauf wird Marketing-Planung zum kritischen Schritt für den geschäftlichen Erfolg. Heute kämpfen Führungskräfte im Bereich Business und Marketing mit schwierigen Entscheidungen: Wie sorge ich dafür, dass meine Firma möglichst optimal online sichtbar ist? Soll ich wie immer auf den gleichen Messen auftreten oder etwa meinen Stand verkleinern und dafür mehr in die Suchmaschinen-Optimierung investieren? Soll ich mehr Content und Videos produzieren oder auf Kanälen wie LinkedIn Anzeigen schalten? Und die Fragen und Zielkonflikte werden nicht weniger. Mit Hilfe einer erprobten und wissenschaftlich fundierten Methodik zur Steigerung der Nachfrage und wachsenden Sales-Pipelines hilft ihnen dieser Workshop, mehr als nur diese Fragen richtig zu beantworten.



Am Ende dieses **Workshops** haben sie:

- Einen priorisierten Plan für Marketing-Investitionen
- Einen Kampagnen-Entwurf und inhaltlichen Marketing-Plan
- Eine Definition der Kunden-Zielgruppe
- Eine Skizze der Inhalte und von spezifischen Topics
- Ausgearbeitete Schlüsselbegriffe
- Eine Vorstellung von der Zweitverwertung von Inhalten
- Handlungspfade für jede Prozessstufe für gesteigertes Engagement
- Kriterien zur Messung des Marketing-Erfolgs

Über Rebecca Geier:

Rebecca Geier verfügt über 25 Jahre an globaler Marketing-Erfahrung für technische und wissenschaftliche Märkte. Von Journalisten des *The Wall Street Journal* wurde sie unter die zehn innovativsten Unternehmer Amerikas gewählt. Rebecca Gaier hat 2016 ihr erstes Buck mit dem Titel *Smart Marketing for Engineers: An Inbound Marketing Guide to Reaching Technical Audiences* veröffentlicht.

Bevor sie als Mitgründerin von TREW Marketing aktiv wurde, hat sie diverse Marketing-Programme für unterschiedliche Technologien und Marketing-Firmen geleitet und war 14 Jahre lang Mitglied des „Marketing Leadership Teams“ von National Instruments. Rebecca Geier hält regelmäßig Vorträge und schreibt Artikel zu Themen rund um effektives Marketing für technische Zielgruppen.

Über das Buch:

Reich an Daten führt das Buch Ingenieure an die Themen online Content, Surf-Verhalten und Surf-Präferenzen heran. Das Buch strotzt von Listen, Vorlagen und Beispielen für B2B-Engineering sowie die wissenschaftlich/technische Industrie inklusive den Märkten Automotive, Verteidigung, Luftfahrt, Consumer-Elektronik, Fertigungstechnik, IT und Energie. Falls Sie als Ingenieur oder technischer Geschäftsführer tätig sind und sich für eine unkomplizierte aber wissenschaftlich fundierte Anleitung für modernes Marketing interessieren oder falls Sie als Marketing-Profi neue technische Zielgruppen ins Visier nehmen, dann ist dieses Buch genau richtig für Sie.

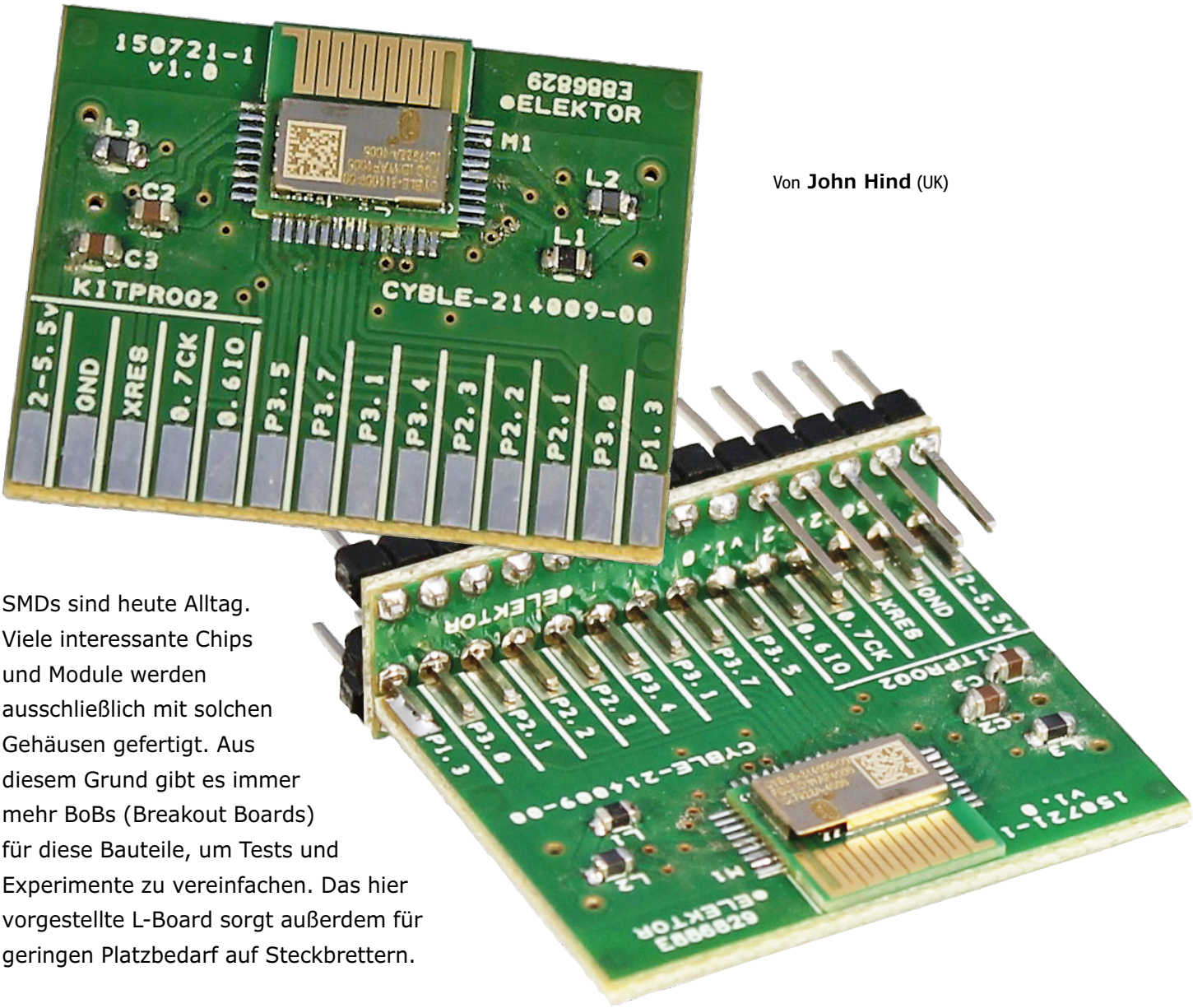
WAS: 1,5-tägiger Marketing-Workshop
WER: Führungskräfte und Teams im Bereich Business und Marketing, die auf technische Zielgruppen wie Ingenieure und Wissenschaftler ausgerichtet sind
WANN: Juni 2017
WO: Berlin, München, Aachen, Eindhoven

Weitere Informationen:
www.elektormagazine.de/smart-marketing

PSoC-BLE-Modul im L-Board-Format

Ideal fürs Prototyping – passend fürs Steckbrett

Von **John Hind** (UK)



SMDs sind heute Alltag. Viele interessante Chips und Module werden ausschließlich mit solchen Gehäusen gefertigt. Aus diesem Grund gibt es immer mehr BoBs (Breakout Boards) für diese Bauteile, um Tests und Experimente zu vereinfachen. Das hier vorgestellte L-Board sorgt außerdem für geringen Platzbedarf auf Steckbrettern.

Normalerweise führt man bei BoBs die winzigen SMD-Kontakte zu Pins im gewohnten Rastermaß 1/10'', das gemeinhin für Stiftleisten und bei Steckbrettern verwendet wird. Das klappt gut bei relativ wenig Kontakten, etwa bei Sensoren und Aktoren. Hier reicht für gewöhnlich eine einzige Pin-Reihe, und man kann so ein Modul mit seiner Stiftleiste direkt in einem Steckbrett verwenden. Ein gutes Beispiel hierfür sind die BoBs, die in der Artikelreihe „Sensoren" in Elektor [1] verwendet wurden. Mit zunehmender Kontaktzahl würden aber einreihige Stiftleisten zu lang werden und sich nicht mehr gut für Steckbretter eignen. Zweireihige Stiftleisten entspannen die Situation etwas, weswegen sie auch für die diversen T-Board-Projekte verwendet wurden, die in Elektor über

Eigenschaften

- Platzsparendes und steckbrettkompatibles DIL-Modul
- PSoC mit BLE 4.1
- Graphisches Programmier-Interface

die Jahre erschienen sind. Doch kann diese Anordnung bei Steckbrettern mit zentralem Steg (für DIL-ICs) problematisch werden. Und wenn das Board mehr als 10 mm breit ist, können dadurch Löcher abgedeckt werden, was den verfügbaren Platz auf dem Steckbrett für andere Bauteile einschränkt. Außerdem braucht es auch bei einem 10 mm breiten Board Pads auf beiden Seiten, wodurch lediglich etwa 6 mm für die Bestückung übrigbleiben, was für viele SMDs nicht ausreicht. Dies gilt auch dann, wenn wie beim T-Board-Format mehrere Leitungen zur Seite geführt werden. Am Ende erzwingt dies oft Boards, die breiter sind als das 10-mm-Ideal.

L-Board-Vorteile

Das hier vorgeschlagene L-Board-Format ist eine Lösung für diese Probleme. Es handelt sich um ein zweiteiliges Board, bestehend aus einem 10 mm breiten Basis-Platinchen, das an eine vertikale, doppelseitige Träger-Platine für den/das eigentliche(n) Chip/Modul gelötet wird. Diese Struktur passt prima in ein Steckbrett und verdeckt wie ein normales DIL-IC keine Steckplätze. Man kann auch prima mehrere L-Boards nebeneinander stecken, wie das bei DIL-ICs auch möglich ist. Dafür muss man – wie bei den bekannten schwedischen Möbeln üblich – so ein L-Board erst selbst zusammenbauen.

Das erste L-Board

Das PSoC-BLE-Modul von Cypress eignet sich sehr gut für das L-Board-Konzept. PSoC steht für Programmable System on Chip. In diesem Fall wird ein leistungsfähiger 32-bit-ARM-Core mit einem BLE-Funk-Modul (Bluetooth Low Energy), etwas programmierbarer Logik und analoger Peripherie samt Schalt-Logik zu einem Ganzen kombiniert. Es gibt außerdem eine ganze PRoC-Familie (Programmable Radio on Chip). Die BLE-PSoC-Chips bieten ein Superset der PRoC-Funktionen, denn die PRoC-Module haben keine programmierbare Logik-Peripherie. Aus Entwicklersicht sind sie ansonsten weitgehend austauschbar. Das letzte „C“ im Kürzel lässt auf die Verfügbarkeit in Chip-Form schließen. Es gibt aber auch komplette, qualifizierte Mikro-Module mit integrierter Antenne und Quarzen, die für kleine Produktionsstückzahlen prima geeignet sind. Aber auch diese Module sind ohne professionelle Lötausrüstung für SMDs kaum von Hand zu löten.

Diese PSoC/PRoC-Technik überzeugt dank unterstützender Entwicklungsmodule und einer kostenlosen IDE durch niedrige Einstiegskosten. Für den Anfang benötigt man lediglich eines dieser Entwicklungs-Kits, die Kitprog2-USB-Programmierung und Debugging unterstützen. Diese Kits kosten unter 10 € – man muss lediglich darauf achten, dass einige dieser Kits lediglich einen USB-Bootloader und nicht den Kitprog2-Adapter beinhalten.

Bild 1 zeigt die Schaltung des ersten

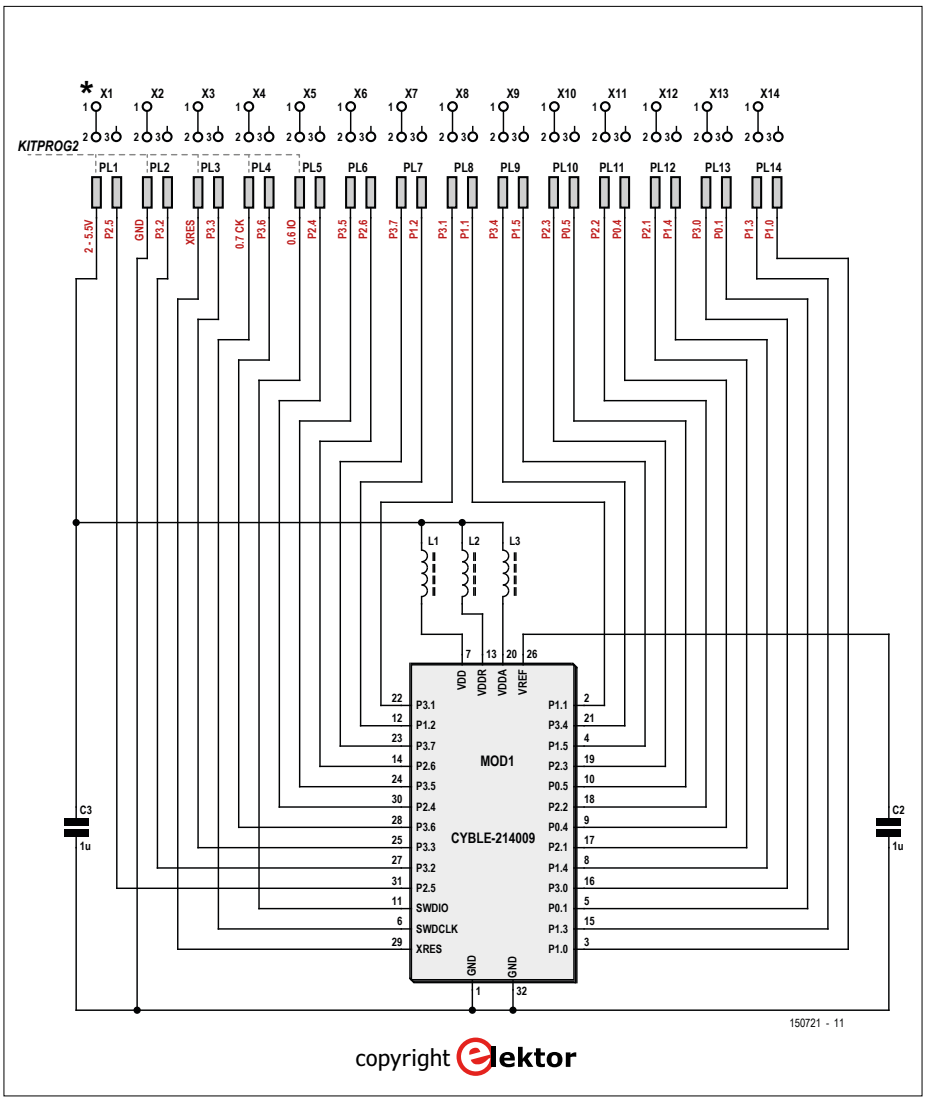


Bild 1. Schaltung des L-Boards für das PSoC-Modul CYBLE-214009-00 mit BLE 4.1. PL1...PL14 bezeichnet den Edge-Connector des Boards; X1...X14 sind die drei Stiftleisten des Basis-Boards.

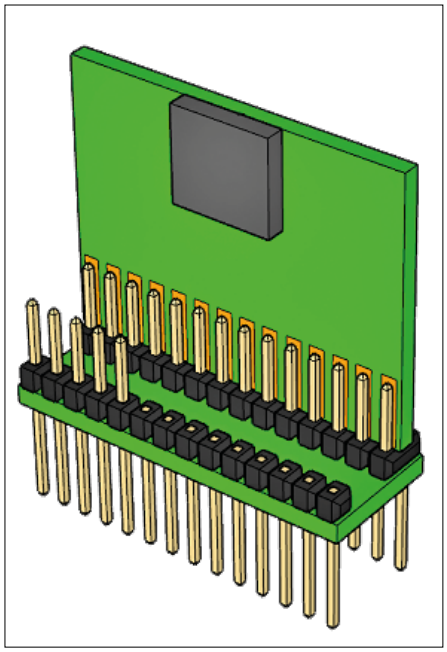


Bild 2. 3D-Rendering des L-Boards. Nicht sichtbar sind die Pins, welche die Rückseite des vertikalen Boards mit dem Basis-Board verbinden.

vorgestellten L-Boards. Als BoB enthält es nicht allzu viele Bauteile. Außer einigen Entkopplungs-Cs und einigen Ferrit-Spulen zur Entstörung sieht man nur das IC (und die Stiftleiste).

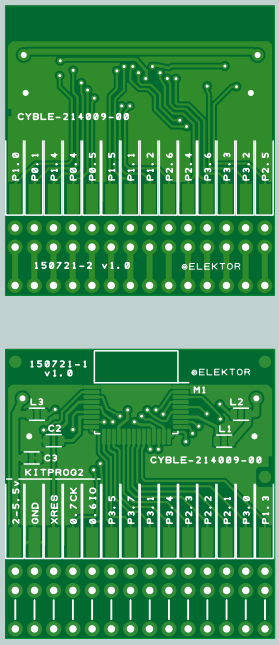
Ideal wäre ja die Verwendung des erweiterten und BLE-4.2-unterstützenden Moduls CYBLE-224116-01, von dem momentan erste Samples verschickt werden. Leider kam das aber für diesen Artikel zu spät, weshalb hier das BLE-4.1-Modul CYBLE-214009-00 zum Zuge kam. Hoffentlich kann hier bald ein Upgrade folgen, denn BLE 4.2 bietet mehr Einsatzmöglichkeiten jenseits des Ersatzes von Kabeln oder eines PAN (Personal Area Network). Auf den Elektor-Labs-Webseiten zu diesem Artikel [2] ist sogar schon eine Platine für dieses neuere Modul verfügbar.

L-Board-Aufbau

Das Board besteht aus zwei Teilen, dank eines sogenannten V-Cut kann es sauber in die beiden Teile gebrochen werden. Die abschließende Ikea-Technik besteht darin, die beiden 28- und 14-poligen Stiftleisten (für diese 28-Pin-Ausführung) auf das kleine Basis-Board zu löten, hierzu schaue man sich Bild 2 an. Die hintere DIL-Stiftleiste erfordert Pins auf beiden Platinenseiten, also Ober- und Unterseite – der vordere Header hat seine Pins nur auf der Unterseite. Die überflüssigen überstehenden Pin-Enden kann man abzwicken. Beim DIL-Header macht man das aus Platzgründen auch bei der vorderen Reihe der unten überstehenden Pins, bevor der vordere Header eingelötet wird. Beim vorderen Header lässt man links oben fünf überstehende Pins stehen, da hier der Kitprog2-Adapter angeschlossen wird. Zum Schluss wird das Modul-tragende vertikale Board mit seinen Pads (wie in Bild 2 zu sehen) zwischen die zwei oberen Pin-Reihen hinten eingelötet.

STÜCKLISTE

C1,C2 = 1µ, 0805
L1..L3 = Ferritspule, 300 Ω @ 100 MHz, 0805
MOD1 = CYBLE-214009-00
28-pol. DIL-Stiftleiste, 2x14, RM 0,1", Pins ~15 mm lang
14-pol. SIL-Stiftleiste, 1x14, RM 0,1", Pins ~15 mm lang
Platine 150721-1



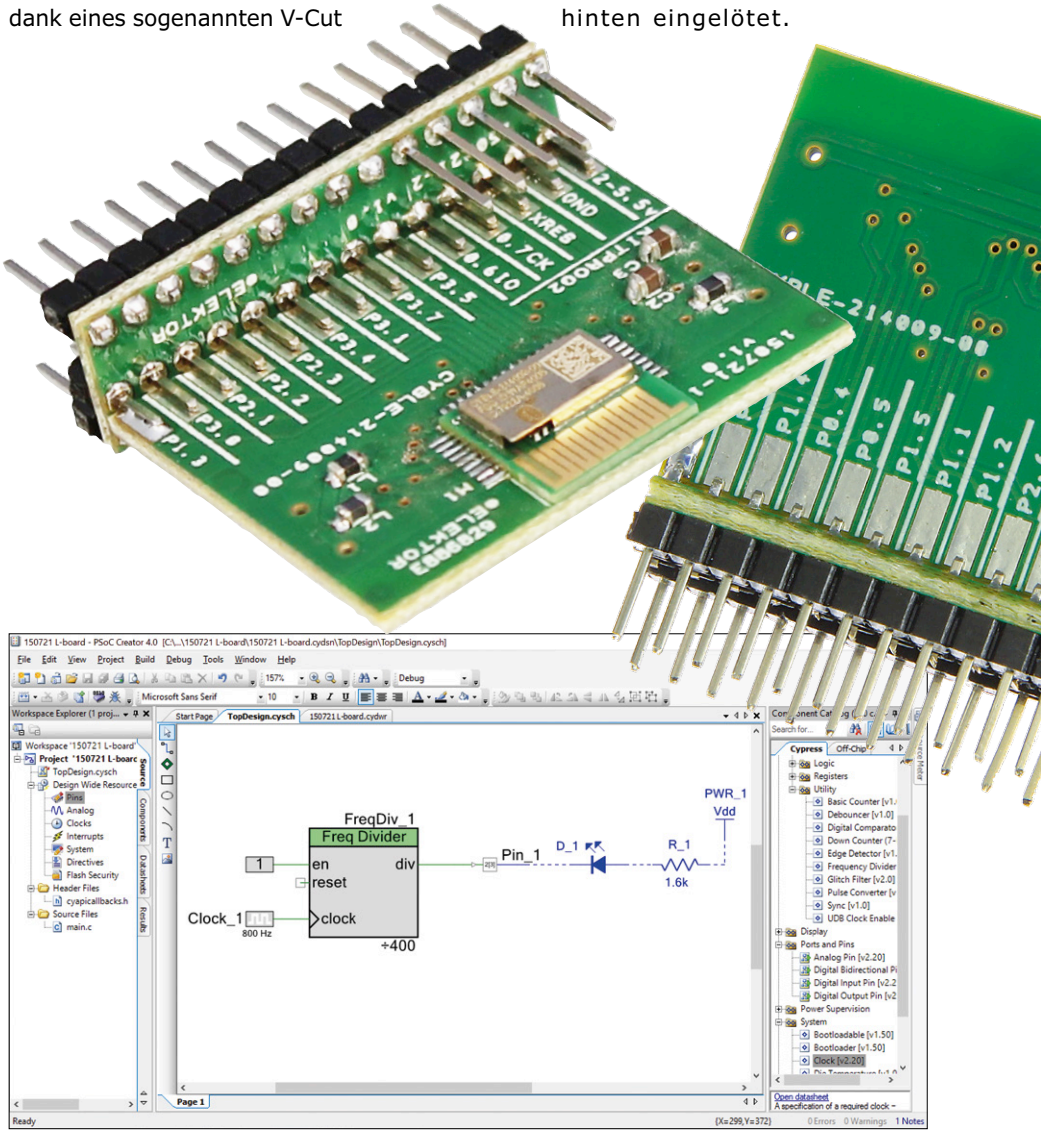


Bild 3. Das „Hello World!" in der Embedded-Version mit PSoC-Creator benötigt nicht einmal den Mikrocontroller. Die blauen Schaltungsteile müssen auf das Steckbrett.

Jetzt passt das resultierende L-Board auf ein Steckbrett.

Hello World!

Vor der Programmierung muss man zuerst die PSoC-Creator-IDE von der Cypress-Webseite herunterladen und auf einem Windows-PC installieren. Jetzt verbindet man das Kitprog2-Flachbandkabel mit den fünf Pins links auf dem Basis-Board (siehe Kasten Programmier-Adapter). Dann wird Kitprog2 per USB mit einem PC verbunden (hierzu ist ein USB-A-Verlängerungskabel ganz nützlich). Auf der „Start Page“ des PSoC-Creators wählt man „Create New Project“. Als „Target module“ wählt man dann „CYBLE-214009-00“. Auf der nächsten Seite wählt man „Empty schematic“ und auf der letzten Seite einen passenden Ordner zur Ablage der Dateien. Es mag überraschen, dass man jetzt tatsächlich auf eine leere Seite blickt statt auf den gewohnten Code-Editor.

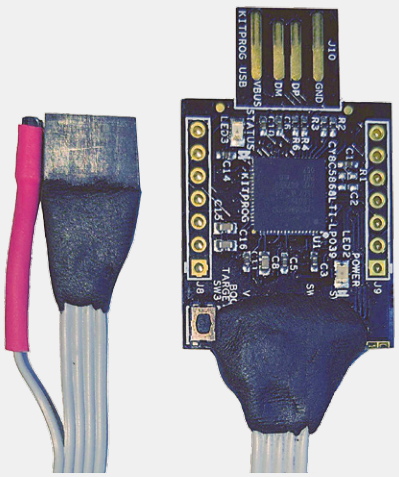
Rechts sieht man den „Component Catalog“. Man ziehe hier einen „Digital Output Pin“ vom Bereich „Ports and Pins“ in die Schaltung, mache einen Rechtsklick darauf und wähle „Configure“. Im Tab „General“ wählt man die Optionen „Digital Output“, „HW connection“ und „External terminal“ und abschließend kommt noch ein Klick auf „OK“. Nun wählt man noch eine LED, einen Widerstand und eine Power-Komponente im Tab „Off-Chip“ im „Component Catalog“ und arrangiert und verbindet sie anhand von Bild 3. Im Workspace links von der Schaltung kommt noch ein Doppelklick auf „Pins“ bei „Design Wide Resources“. Man sieht ein Bild mit der Pin-Belegung des Moduls und einer Liste der Pins der Schaltung, die im Moment lediglich aus einem vorher hinzugefügten Ausgangs-Pin besteht. Im Drop-Down-Menü „Port“ verknüpft man diesen mit einem real existierenden Port. Jeder Port wäre möglich, aber hier wählt man zunächst „P2[3]“. Auf dem L-Board ist dieser Pin mit P2.3 beschriftet. Natürlich muss man jetzt noch die LED-Schaltung auf dem Steckbrett aufbauen. Die Bauteile außerhalb des Chips sind nur zu Dokumentationszwecken in der Schaltung eingezeichnet.

Mit einem Klick auf den Tab „TopDesign.cysch“ landet man wieder bei der Schaltung. Via Tab „Cypress“ des „Component Catalog“ fügt man das Bauteil „Clock“ hinzu (steckt im Verzeichnis „System“) und setzt die Frequenz bei seinen Properties auf 800 Hz. Das ist nahe an der minimalen Taktfrequenz, die mit Hilfe der eingebauten Frequenzteiler möglich ist. Dazu kommt noch das Bauteil „Frequency Divider“ (im Verzeichnis „Digital\Utility“), und man stellt es auf einen Teiler von 400. Der Clock-Anschluss kommt natürlich an Clock und sein Ausgang „div“ an den Pin. Zum Schluss verbindet man noch ein „Logic High“ (im Verzeichnis „Digital\Logic“) mit dem Pin „Enable“ des Teilers. Das Resultat sollte Bild 3 deutlich ähneln. Ist das der Fall, sichert man seine Arbeit.

Nun noch die Option „Program“ im Menü „Debug“. Nachdem dies erledigt und das Ergebnis übertragen wurde, sollte die LED direkt anfangen zu blinken. Die blinkende LED ist das embedded Äquivalent von „Hello World!“ für Software – diesmal ohne eine einzige Zeile Code. Tatsächlich ist nicht einmal das ARM-SoC aktiv, denn dieses Beispiel benötigt nur die programmierbare Logik! Nun füge man noch ein „Control Register“

Programmier-Adapter

Für einen universellen Programmier-Adapter trennt man den Kitprog2-Teil von einem entsprechenden Kit-Board ab (ist in der Regel perforiert) und lötet ein Stück fünfpoliges Flachbandkabel in die fünf Pin-Löcher auf der Platine gegenüber dem USB-Stecker ein. Das andere Ende des Flachbandkabels erhält eine fünfpolige SIL-Buchsenleiste. Am besten trennt man das Kabel (und die Buchse) ein Stück weit in eine Leitung für VTARG und vier gemeinsame Leitungen für die restlichen Signale auf. Damit kann man das Board entweder von den 5 V der USB-Buchse oder aber getrennt z.B. von der Spannung eines Steckbretts versorgen (ein Vorteil der PSoC-Architektur ist ihr breiter Versorgungsspannungsbereich von 1,9...5,5 V). Man kann dann auch eine fünfpolige Stiftleiste auf das ursprüngliche Board des Kits löten und dieses dann ebenfalls mit dem nun separaten Kitprog2 betreiben. Das Foto zeigt ein fertiges Kitprog2 mit einem angelöteten und per „Sugru“ verklebten Kabel. Sugru eignet sich sehr gut für allerlei Isolierungszwecke in der Elektronik.



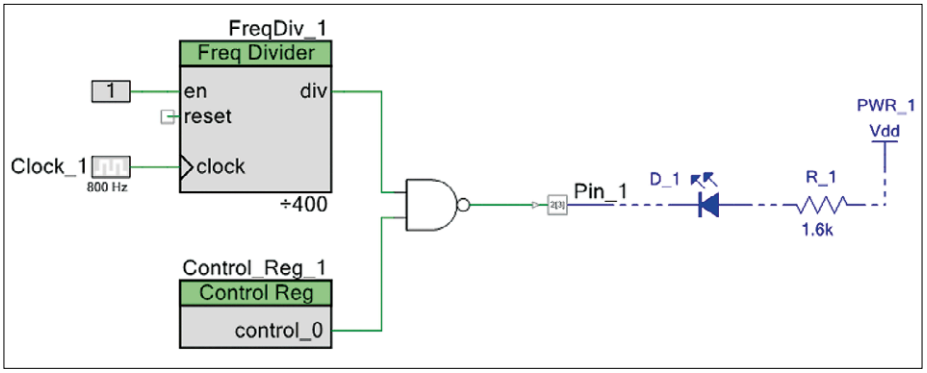


Bild 4. In den erweiterten Versionen von „Hello World!“ wird die LED vom Mikrocontroller und der programmierbaren Logik gesteuert.

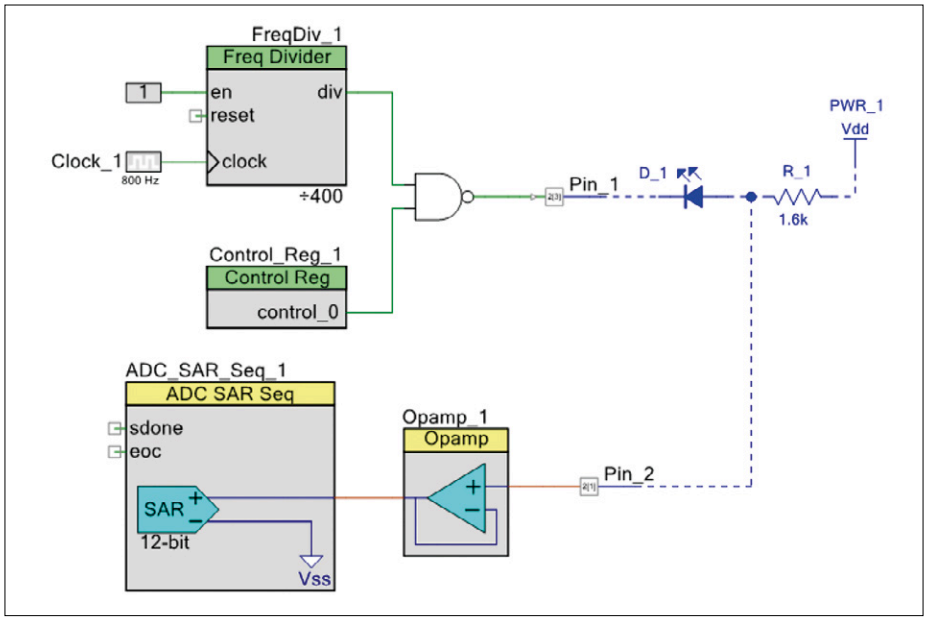


Bild 5. Durch Hinzufügen eines ADC-Blocks kann man die Flussspannung einer LED messen.

Listing 1. Ein einfaches Test-Programm für das Debugging und ein Single-Stepping durch den Code.

```
#include "project.h"

int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    for(;;)
    {
        Control_Reg_1_Write(0x01);
        Control_Reg_1_Write(0x00);
    }
}
```

Listing 2. Das Programm für die Schaltung in Bild 5.

```
#include "project.h"

int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    volatile float32 volts = 0;
    ADC_SAR_Seq_1_Start();
    ADC_SAR_Seq_1_StartConvert();

    for(;;)
    {
        Control_Reg_1_Write(0x01);
        volts = ADC_SAR_Seq_1_CountsTo_Volts(0,
                  ADC_SAR_Seq_1_GetResult16(0));
        Control_Reg_1_Write(0x00);
    }
}
```

plus ein „NAND"-Gatter zur Schaltung hinzu und setze die Anzahl der „Outputs" auf „1". Die Verbindungen werden anhand von Bild 4 vorgenommen. Links im „Workspace Explorer" öffnet man die Datei „main.c" und ergänzt sie um zwei Code-Zeilen in der „for"-Schleife (siehe Listing 1):

```
Control_Reg_1_Write(0x01);
Control_Reg_1_Write(0x00);
```

Nun wählt man „Debug" aus dem gleichnamigen Menü. Dies startet das ARM-SoC und stoppt dessen Prozessor vor der Ausführung der ersten Code-Zeile. Drückt man jetzt zweimal F10, beginnt die LED zu blinken. Drückt man weiter auf F10, stellt man fest, dass die LED aufleuchtet, wenn das Control-Register gesetzt wird, und erlöscht, wenn das Register gelöscht wird.

Messen einer Spannung

Zur Untersuchung der analogen Fähigkeiten kann man z.B. die Flussspannung einer LED messen. Hierzu geht man auf die Seite „System" der „Design Wide Resources" und stellt die Versorgungsspannung auf „5.0 V" (in der Annahme, dass das Board von Kitprog2 versorgt wird). Man füge einen „Analog"-Pin, einen „Opamp" und einen „Sequencing SAR ADC" zur Schaltung hinzu. Beim Pin wird ein „External terminal" konfiguriert, der Opamp wird „Follower" und der ADC erhält eine Sample-Rate von 100.000 S/s. Sein „Vref select" wird auf „VDDA", seine „Sequenced channels" auf „1", sein „channel 0" im Tab „Channels" wird auf „enabled" und sein „Mode" auf „Single" eingestellt. Mit dem Tab „Pins" wird der Pin mit „P2[1]" assoziiert. Die Verdrahtung erfolgt anhand Bild 5 – dabei erfolgt die Verbindung der LED mit dem analogen Pin physikalisch auf dem Steckbrett! Mit ein paar Zeilen Code wird aus Listing 1 jetzt Listing 2:

```
volatile float32 volts = 0;
ADC_SAR_Seq_1_Start();
ADC_SAR_Seq_1_StartConvert();
```

Im Block „for" kommt nach den beiden Schreib-Instruktionen des Control-Registers jeweils folgende Zeile:

```
volts = ADC_SAR_Seq_1_
    CountsTo_Volts(0,
    ADC_SAR_Seq_1_GetResult16(0));
```

Man debugge und beobachte die Variable „volts“. Abhängig von Typ und Farbe der LED sollte sie Werte zwischen 1,8 und 4,5 enthalten. Leuchtet die LED gerade nicht, sollte ein Wert von 0 V zu sehen sein. Im Tab „Analog“ von „Design Wide Resources“ kann man sehen, wie der analoge Teil des Chips implementiert ist. Auch ein Blick in „Resource Meter“ (via dem Tab oben rechts) lohnt sich. Hier kann man darüber informiert werden, dass allein für die blinkende LED schon fast die Hälfte der UDB-Ressourcen verbraucht wird. Andere Methoden wären, eine der vier „Timer Counter PWM“-Einheiten oder ganz traditionell einen Timer-Interrupt samt Code zu verwenden.

+ Bluetooth

Bluetooth Low Energy wird in einem als Download erhältlichen Projekt [2] behandelt. Ein kurzer Blick zu BLE: Man hole die Komponente „Bluetooth Low Energy“ von der Sektion „Communications“ des Katalogs und öffne ihren Konfigurations-Dialog. Man kann jetzt eines der vielen geprüften Standard-Profile auswählen oder aber mit „Custom“ ein neues Profil anlegen. Gewöhnlich wählt man GATT-Server und GAP-Peripherie (Verbindung mit Smartphones, Tablets oder PCs). Die Charakteristik und die Deskriptoren werden im Knoten „Custom Service“ auf der Seite „Profiles“ definiert. Das Modul kann eine zentrale Rolle spielen oder dynamisch zwischen Rollen umschalten. Dies ermöglicht Punkt-zu-Punkt-Verbindungen oder Netzwerke mit mehreren Modulen ohne Smartphone, Tablet oder PC. Der Download bietet mehr Details und generischen Code für einen eigenen Dienst sowie ein voll ausgearbeitetes Beispiel.

Ein Blick in die EZ-Serial-Firmware von Cypress lohnt sich vor allem, wenn man nicht so fit in der C-Programmierung ist. Die freie CySmart-App für Android, iOS und Windows ist ein tolles Tool für das Debugging und den Test von BLE (Windows-PCs brauchen einen geeigneten USB-Dongle).

Fazit

Diese Einführung in das L-Board-Format demonstriert seine Vorteile gegenüber dem etablierten T-Board-Format und anderen „flachen“ BoBs. Es wurden ein modernes BLE-PSoC-Modul ins L-Board-Format gebracht und damit einige Experimente zwecks Demonstration von grundlegenden Eigenschaften beschrieben. Von Cypress gibt es aktuell erste Samples einer neueren Modul-Version mit dank BLE 4.2 verbesserter Kapazität, Sicherheit und Leistung sowie mehr Anwendungsmöglichkeiten. Wenn die neue Version dann lieferbar ist, kann man auch diese Module in das L-Board-Format bringen. Die Verbesserungen eröffnen für BLE interessante neue Applikationen jenseits von reinen Funkverbindungen oder kleinen Netzwerken. Der Bereich Heimautomation und möglicherweise neue, preiswerte Geräte für das Labor etc. bieten sich an.

(150721)

EINKAUFSZETTEL

→ 150721-1
Platine

Weblinks

[1] www.elektormagazine.de/magazine/elektor-201611/39894

[2] www.elektormagazine.com/labs/l-board

[3] www.elektormagazine.de/150721

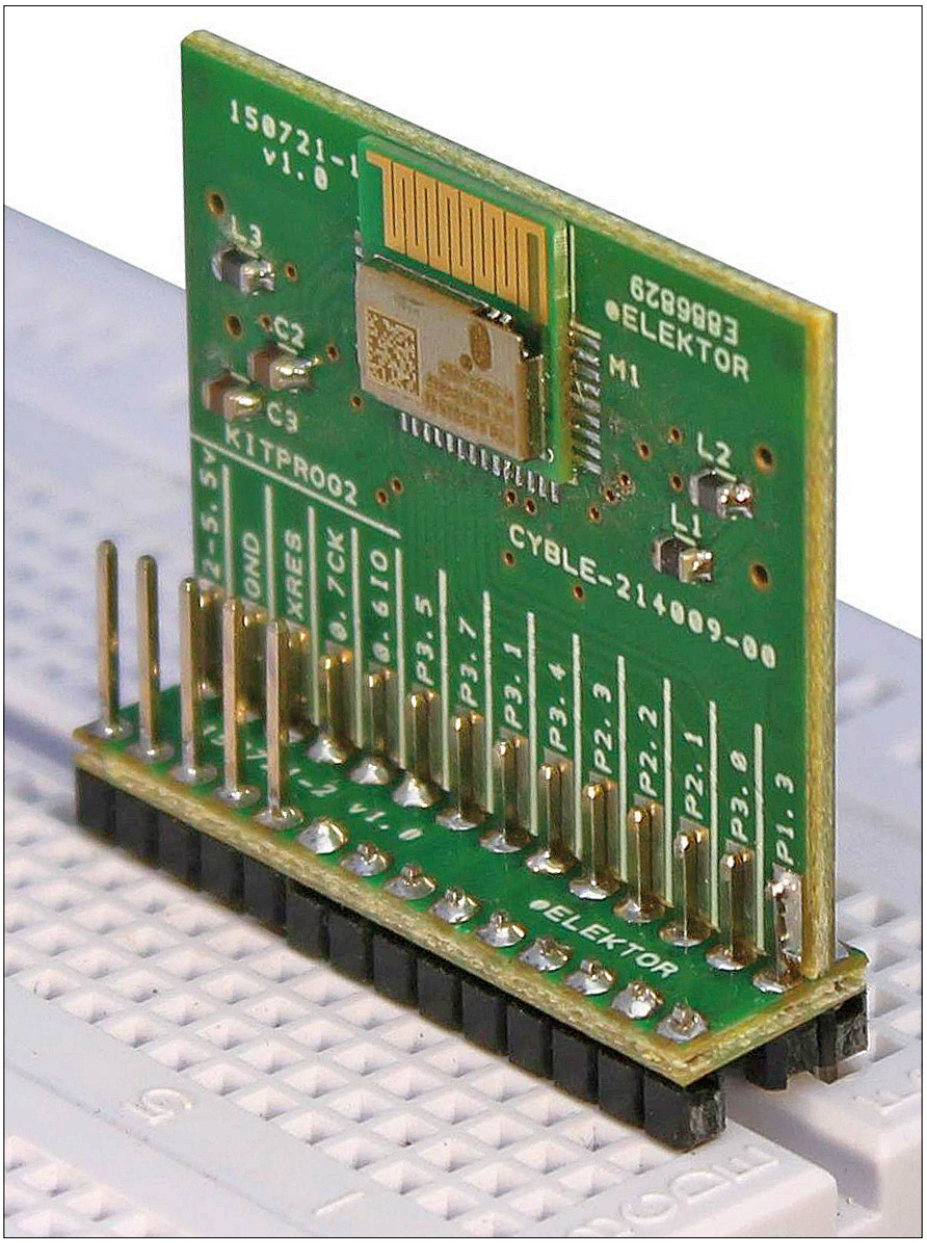


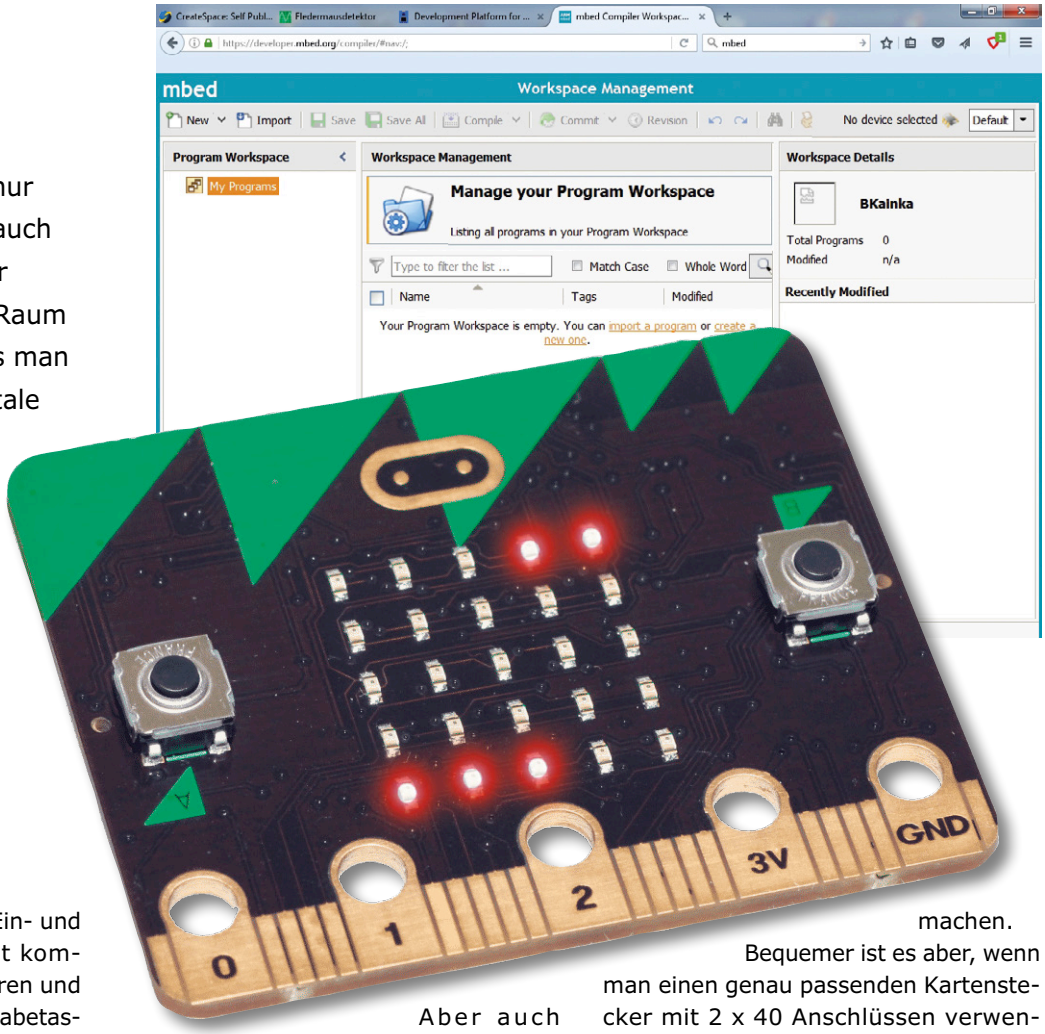
Bild 6. Fertiges L-Board auf einem Steckbrett.

BBC micro:bit für Elektroniker

Einstieg mit mbed

Von **Burkhard Kainka**

Der BBC micro:bit ist nicht nur für Schüler geeignet, er ist auch ein tolles Controllerboard für Elektroniker. Auf kleinstem Raum ist fast alles vorhanden, was man immer wieder braucht: Digitale Ein-/Ausgänge mit PWM, AD-Wandler sowie eine USB-Schnittstelle für Stromversorgung, Programmierung und Datenaustausch. Verschiedene Sensoren und Bluetooth runden die Ausstattung ab.



Auch ein Arduino bietet diverse Ein- und Ausgänge, beim BBC micro:bit kommen aber noch zahlreiche Sensoren und Bedienelemente hinzu. Zwei Eingabetaster und ein 5x5-LED-Anzeigefeld, ein Kompass-Sensor und ein 3-Achsen-Beschleunigungssensor, dazu Licht- und Temperaturmessung. Und als Krönung gibt es die Datenübertragung mit Bluetooth Low Energy.

Kontaktierung

Die Platine wurde von der BBC in Zusammenarbeit mit der Universität Lancaster speziell für den Einsatz in Schulen ausgelegt. Daher gibt es hier fünf große Anschlusspunkte mit 4-mm-Löchern, an die man ganz einfach Krokodilklemmen anschließen kann (**Bild 1**). Prinzipiell besteht zwar die Gefahr, dass man dabei auch die kleineren Nachbarpins berührt. Aber auch daran wurde gedacht. Die Nachbarpins der Anschlüsse GND und 3 V haben das gleiche Potential. Und neben den großen Portanschlüssen 0, 1 und 2 findet man ebenfalls Portpins, die eine direkte Verbindung klaglos überstehen. Die ersten einfachen Versuche kann man tatsächlich mit Krokokabeln durchführen (siehe Titelbild). Wenn es aber komplexer wird und man mehr Anschlüsse benötigt, muss irgendeine Art von Stecker her, wenn man nicht einfach Drähtchen anlöten will. Die Platine passt übrigens mit ihren Anschlüssen im 1,27-mm-Raster in die Karten-Slots eines ausrangierten PC-Motherboards. Mit Zange, Säge und Lötkolben lässt sich da sicher etwas machen.

Bequemer ist es aber, wenn man einen genau passenden Kartenstecker mit 2 x 40 Anschlüssen verwendet. Geeignete Platinen gibt es im Elektor-Shop [6][7]. **Bild 2** zeigt die Platine des Projekts „Wetterstation für BBC micro:bit“.

Eine Alternative ist die Verwendung von doppelreihigen Pfostensteckern mit 2 x 20 Pins, die man einfach an die Kontakte anlötet (**Bild 3**). Die Stifte der Pfostenstecker haben allerdings den doppelten Pinabstand von 2,54 mm. Deshalb erwischt man bei den kleinen Anschlüssen des BBC micro:bit leider nur jeden zweiten.

Auf der Rückseite sieht es ähnlich aus. Man beachte aber, dass alle Anschlüsse

auf der Platinen-Rückseite vollkommen isoliert sind. Die dort angelöteten Pins dienen erstmal nur der mechanischen Stabilität. Die Pins auf der Rückseite haben aber auch noch eine Reservefunktion: Wenn doch mal ein bisher nicht angeschlossenes Signal des Micro:bit gebraucht wird, kann einer der hinteren Pins mit einem Drähtchen angeschlossen werden. Dazu muss man aber für den Einsatz auf einer Steckplatine den entsprechenden Pin in der vorderen Reihe entfernen, damit es keinen Kurzschluss gibt. Nicht benötigte Pins gibt es ja genug. Da sind zum Beispiel die zusätzlichen Anschlüsse für GND und VCC. Was aber mal wichtig werden kann, sind die I2C-Anschlüsse, die man mit der Stiftleisten-Methode nicht alle erreicht. Deshalb wird der überflüssige Stift beim GND-Anschluss entfernt. Das geht sehr gut: Mit dem Lötkolben heiß machen, dann rausziehen. Durch die Lücke kann dann ein Drähtchen gefädelt werden, das an den gewünschten Anschluss (SDA, P20) gelötet wird. Auf der Rückseite wird das Drähtchen dann an den entsprechenden Pin angelötet. Und schon ist der I2C-Bus gerettet.

Die ersten Programme

Praktische Anwendungen im Bereich der Elektronik sind ganz einfach zu realisieren. Es gibt dazu viele Programmiersprachen, die in erster Linie für den Einsatz in Schulen entwickelt wurden [1][5]. Hier soll aber mit C++ gearbeitet werden, weil damit praktisch alles möglich ist, während andere Sprachen immer gewisse Einschränkungen haben. Mit der Mbed-Plattform [2] ist das besonders einfach, weil man online arbeitet und nichts installieren muss.

Melden Sie sich bei Mbed an und richten Sie sich einen eigenen Bereich ein, falls Sie das nicht schon für andere Projekte getan haben. Wenn Sie nun versuchen, ein vorhandenes Beispiel zu laden, werden Sie daran erinnert, eine Plattform zu wählen, also ein System mit dem sie arbeiten wollen. Sie werden auf die Hardware-Seite verwiesen und können erstmal staunen, wie viele Boards da schon verwendbar sind, eins davon übrigens von Elektor. In diesem Fall ist der Micro:bit das Zielsystem. Hier stößt man auf weitere Informationen wie z.B. das Anschlussbild des Micro:bit und auf die Schaltfläche „add to your mbed Compiler“. Außerdem gibt es hier die entschei-

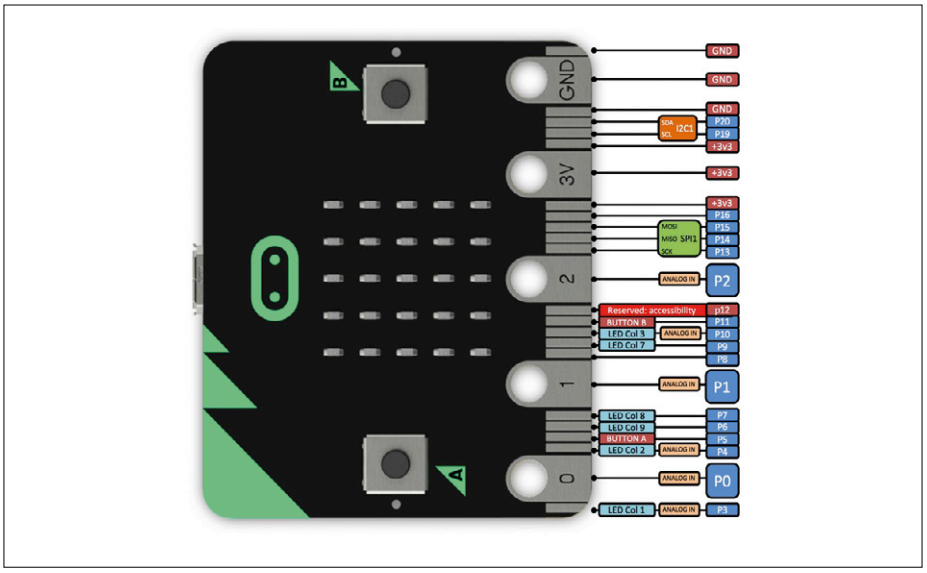


Bild 1. Anschlüsse des BBC micro:bit.

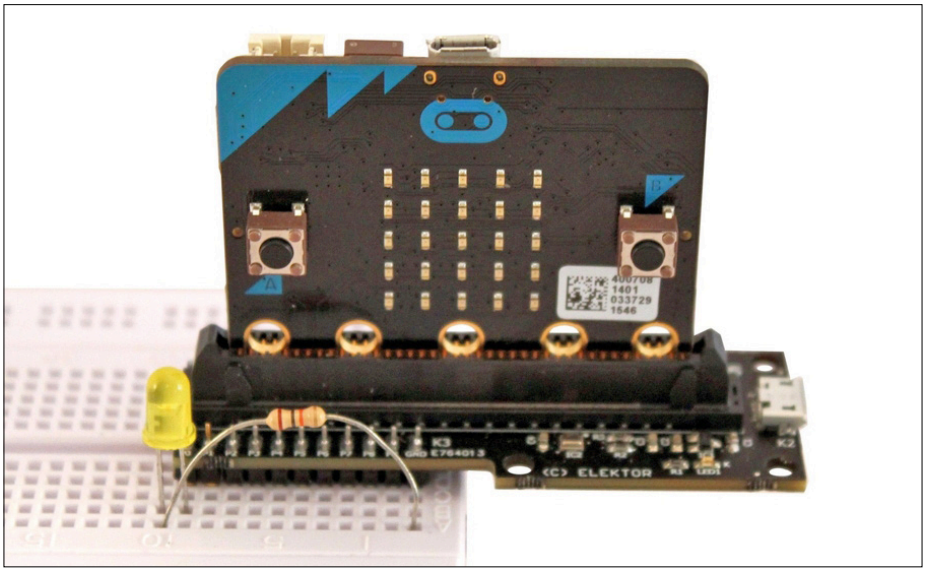


Bild 2. Elektor-Platine 150652-1 als Experimentier-Plattform.

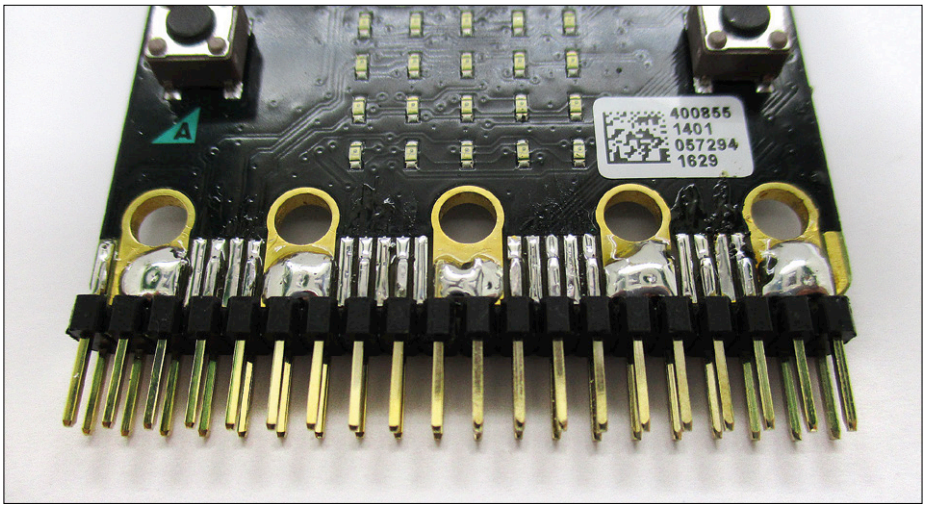


Bild 3. Verwendung von Pfostensteckern.

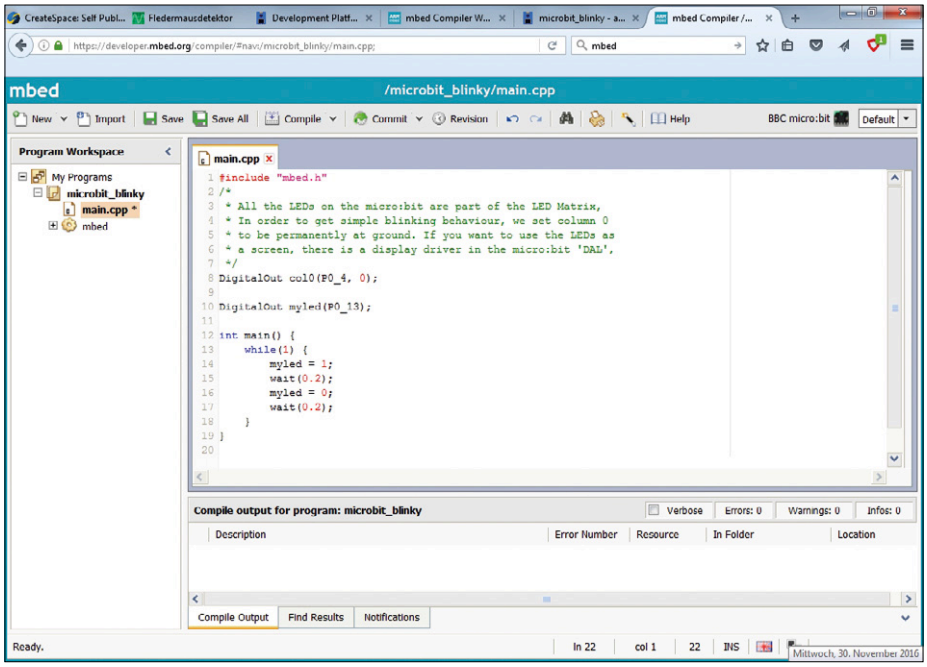


Bild 4. Das Projekt microbit_blinky.

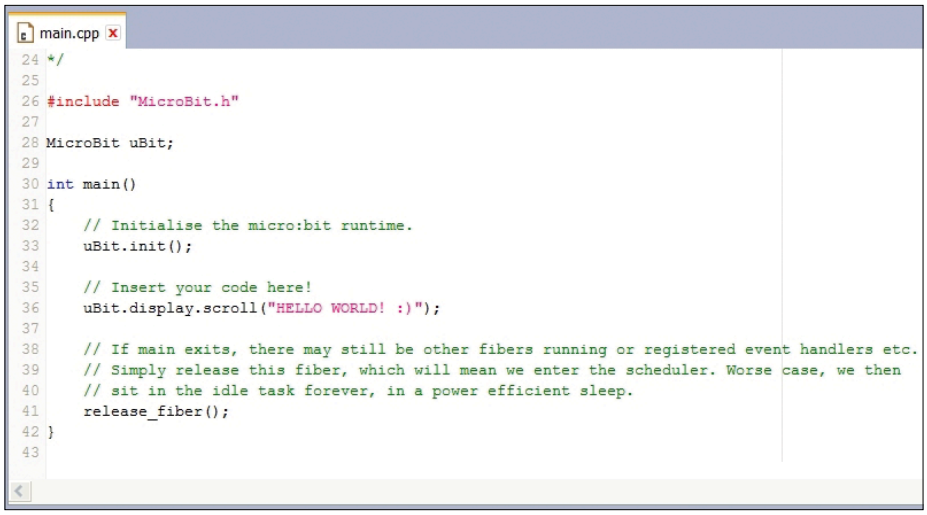


Bild 5. Das Projekt microbit-hello-world.

Listing 1. Spannungsmessung.

```
//Voltage1
#include "MicroBit.h"

MicroBit uBit;

int main()
{
   uBit.init();
   MicroBitSerial serial(USBTX, USBRX);
   while (1) {
        int u = 3300 * uBit.io.P0.getAnalogValue()/ 1023;
        uBit.display.scroll(u);
        uBit.serial.printf("%d\r\n", u);
        uBit.sleep(500);
        }
}
```

denden Links auf die Dokumentation der Lancaster-Universität und erste Beispiele. Das erste Beispiel heißt microbit_blinky und soll nun geladen werden. Im Compiler sieht man die neu eingefügte Plattform und das importierte Programm. Ein Klick auf main.cpp öffnet den übersichtlichen Quelltext (**Bild 4**).

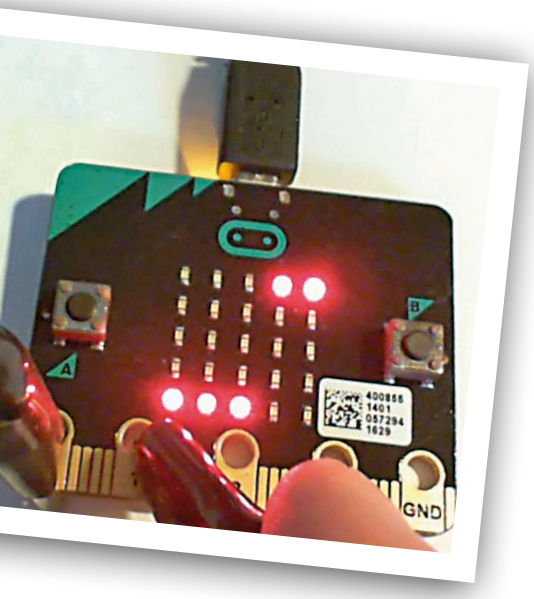
Was sofort auffällt ist die Bezeichnung der Portanschlüsse mit P0_4 und P0_13, also ganz anders als in der offiziellen Anschlussbelegung. Das liegt daran, dass man mit dem Einfügen von mbed.h zwar den Controller auf dem Micro:bit anspricht, aber noch nicht das Board mit allen seinen Eigenschaften und Möglichkeiten. Aber das Programm kann nun getestet werden. Mit Compile erzeugt man ein Hex-File, das dann einfach in das USB-Speichermedium des Micro:bit kopiert wird. Sofort beginnt die gelbe Status-LED auf der Platine zu blinken und zeigt, dass das System gerade dabei ist, den Controller zu programmieren. Nach einer Sekunde ist der Vorgang beendet und man kann das Ergebnis sehen: Die obere linke LED auf dem 5x5-LED-Display blinkt.

Das zweite Beispiel heißt microbit-hello-world und bindet microbit.h ein (**Bild 5**). Damit wird es möglich, die ganz spezifischen Möglichkeiten des Boards zu nutzen. Das Importieren des Programms dauert auffallend lange, was darauf hinweist, dass sehr viele Dateien geladen werden. Das ist der große Pluspunkt von Mbed: Alles was man früher mühsam zusammenkopieren und einbinden musste, macht Mbed selbst. Mit einer anderen IDE auf dem eigenen Computer würde man das auch schaffen, aber vielleicht sehr lange daran arbeiten oder sogar vorzeitig aufgeben. Man kann mit Mbed arbeiten, ohne auch nur zu ahnen, was da alles im Hintergrund abläuft. Im Beispiel wird eine Laufschrift erzeugt. Die entscheidende Zeile erklärt sich selbst: uBit.display.scroll(„HELLO WORLD! :)“); Kompilieren, übertragen, läuft.

Zunächst könnte man meinen, nur eine Laufschrift, dafür so ein Aufwand? Aber tatsächlich bietet Mbed viel mehr, nämlich den kompletten Zugang zu allen wichtigen Hardware-Elementen des Systems, also nicht zur zum Display, son-

Listing 2. Anzeige der Sensorwerte.

```
//Sensors
#include "MicroBit.h"
MicroBit uBit;
int main()
{
   uBit.init();
   MicroBitSerial serial(USBTX, USBRX);
   while (1) {
        uBit.serial.printf("Time: %d ms \r\n", uBit.systemTime());
        uBit.serial.printf("Temp: %d deg \r\n", uBit.thermometer.getTemperature());
        uBit.serial.printf("P0: %d mV  \r\n", uBit.io.P0.getAnalogValue());
        uBit.serial.printf("P1: %d mV  \r\n", uBit.io.P1.getAnalogValue());
        uBit.serial.printf("P2: %d mV  \r\n", uBit.io.P2.getAnalogValue());
        uBit.serial.printf("X: %d mG  \r\n", uBit.accelerometer.getX());
        uBit.serial.printf("Y: %d mG  \r\n", uBit.accelerometer.getY());
        uBit.serial.printf("Z: %d mG  \r\n", uBit.accelerometer.getZ());
        uBit.serial.printf("B: %d µT  \r\n", uBit.compass.getFieldStrength());
        uBit.serial.printf("\r\n");
        uBit.sleep(1000);
        }
}
```



dern auch zu den Ports mit AD-Wandler und PWM, zu den speziellen Sensoren wie Kompass und Beschleunigungssensor und zu vielem mehr. Im Workspace gibt es nun das Verzeichnis microbit und darin das Unterverzeichnis microbit-dal (Device Abstraction Layer) mit unglaublich vielen Dateien, durch die man sich stundenlang durchwühlen kann, um alle Möglichkeiten zu entdecken.

Aber leichter hat man es mit der Dokumentation von der Uni Lancaster [3]. Darin findet man zu jedem Thema alle wichtigen Informationen und auch Code-Schnipsel, die man kopieren und in den eigenen Quelltext einfügen kann. Man könnte also einfach den vorhandenen Quelltext main.cpp im Projekt microbit-hello-word erweitern und nacheinander alles ausprobieren. Die Alternative ist, das Projekt zu klonen und dann an der Kopie weiterzuarbeiten. Das geklonte Projekt heißt nun z.B. microbit-test und tut erstmal dasselbe. Im Folgenden werden kurze Quelltexte vorgestellt, die man einfach in die main.cpp kopieren und so testen kann. Alle Programme findet man zum Download im txt-Format auf der Elektor-Website [8].

Spannung messen

Was man im Elektronik-Labor immer brauchen kann sind AD-Eingänge. Der Microbit hat einen 10-Bit-AD-Wandler mit sechs möglichen Eingängen, wovon drei an den großen Anschlusspunkten liegen. Zum Testen der Eigenschaften wird hier ein einfaches Programm vorgestellt (**Listing 1**). Mit uBit.io.P0.getAnalogValue() erhält man einen 10-Bit-Wert für den Spannungsbereich bis 3,3 V. Er wird in mV umgerechnet und dann angezeigt.

Die Micro:bit-typische Methode der Datenausgabe ist die Laufschrift auf dem LED-Display. Das funktioniert auch recht gut, die Schrift ist aber nur mit einiger Konzentration zu lesen. Aber es gibt ja noch die serielle Schnittstelle, sodass man die Daten auch in einem Terminal sichtbar machen kann. Um das Ganze am PC unter Windows zu nutzen, muss man den mbed Windows serial port driver installieren. Wie das genau geht, wird in [4] beschrieben.

Verbindet man den Eingang 0 mit dem 3-V-Anschluss der Platine, wird wie erwartet 3300 mV angezeigt. Vergleichsmessungen zeigen, dass die tatsächliche Spannung geringfügig kleiner ist. Auf der Platine werden nämlich die 5 V vom USB mit einem Spannungsregler auf 3,3 V stabilisiert und dann über eine Schottkydiode an VCC weitergereicht.
Bei offenem Eingang zeigt die Messung etwa 880 mV. Misst man die Spannung gleichzeitig hochohmig mit dem Oszilloskop, dann findet man eine höhere Spannung, die bei jeder Messung kurz einbricht. Tatsächlich gibt es an den drei wichtigsten Anschlüssen 0, 1 und 2 jeweils Pullup-Widerstände von 10 MΩ, damit die digitalen Portpins ohne großen Aufwand als Berührungssensoren dienen können. Die wirkliche Leerlaufspannung sollte also bei 3,3 V liegen. Um diese auch zu messen, kann man einen Kondensator mit 100 nF an den Eingang legen. Er lädt sich dann auf die Leerlaufspannung auf und puffert sie während der Messung.

Alle Sensoren

Das zweite kleine Beispiel (**Listing 2**) zeigt zahlreiche Messwerte der verschiedenen Eingänge und Sensoren. Dazu gehören auch die Temperatur des Controllers und die Systemzeit in Millisekunden seit dem letzten Start. Dazu drei Spannungen an den Pins 0 bis 2, Beschleunigungswerte an drei Achsen und eine XYZ-kombinierte magnetische

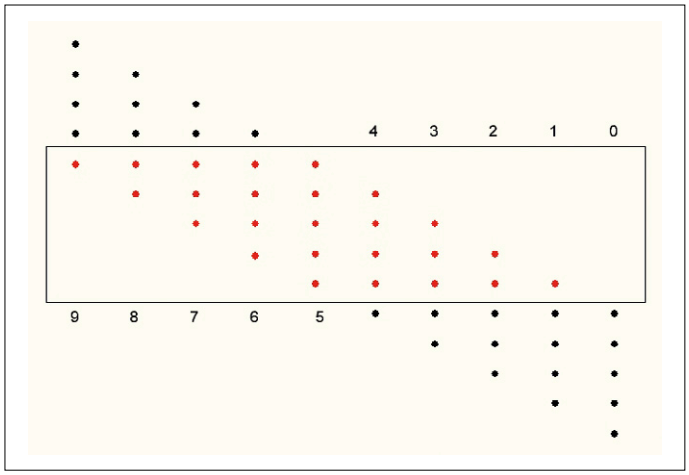


Bild 6. Darstellung der Ziffern.

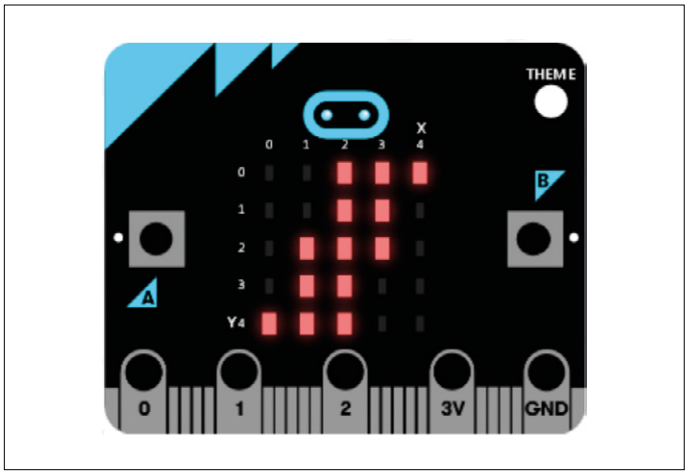


Bild 7. Anzeige der Zahl 13579.

Feldstärke. Dass da zwei komplexe Sensoren über den I2C-Bus ausgelesen werden, nimmt man einfach so hin, andere haben den schwierigen Teil der Arbeit schon geleistet. Nur wenn man es genau wissen will, muss man sich durch die tieferen Schichten zahlloser eingebundener Dateien arbeiten.

Alle Messwerte werden im Terminal angezeigt. Und während man alles ausprobiert, kommen auch die Ideen für mögliche Anwendungen, wie z.B. Messungen an Magneten. Oder wie wäre es mit einem Beschleunigungssensor für das Auto, der einem Fahrer anzeigt, wann er zu sportlich fährt? Die Ausgabe sieht etwa so aus:

```
Time: 1715018 ms
Temp: 22 deg
P0 465 mV
P1 251 mV
P2 252 mV
X 0 mG
Y 1008 mG
Z 64 mG
B: 169311 uT
```

Statisches nummerisches Display

Eine LED-Anzeige mit Laufschrift ist nicht einfach zu lesen, man muss schon sehr konzentriert hinsehen. Wenn man dann eine Ziffer verpasst hat, wartet man auf die nächste Ausgabe. Aber bei 25 LEDs gibt es noch andere Möglichkeiten. Wir sind ja an mehrstellige stehende Anzeigen gewohnt oder alternativ an analoge Zeigerinstrumente.

Eine stehende Anzeige könnte mit Binärzahlen arbeiten oder im BCD-Code. In beiden Fällen braucht man vier LEDs pro Digit. Da aber fünf LEDs in einer Spalte stehen, bietet sich eine andere Darstellung an. Die Ziffern 1 bis 5 werden beginnend von unten durch eine bis fünf LEDs dargestellt. Für die Ziffern 6 bis 9 wandert der Leuchtbalken nach oben. Man kann sich vorstellen, dass man immer fünf Punkte hat, von denen aber nicht immer alle im sichtbaren Bereich liegen (**Bild 6**).

Eine solche Anzeige (**Bild 7**) ist nach etwas Übung intuitiv lesbar. Die Darstellung entspricht übrigens zufällig genau dem Aufbau von Morsezeichen für die Ziffern 0 bis 9, wobei eine leuchtende LED einem Punkt entspricht, eine ausgeschaltete einem Strich. Die Sieben wird also mit --... gemorst.

Für diese Art der Ausgabe wurde eine C-Funktion geschrieben. Die auszuge-

Listing 3. Spannungsmessung mit statischer Anzeige.

```
//LED-Display
#include "MicroBit.h"
MicroBit uBit;

void ledDispay (int n){
    int x;
    int y;
    int d;
    uBit.display.enable();
    MicroBitImage image(5,5);
    image.clear();
    for(int i = 0; i < 5; i++){
            d = 9 - n % 10;
            n = n / 10;
            x = 4 - i;
            for(int j = 0; j < 5; j++){
                  y = d - j;
                  image.setPixelValue(x,y,255);
            }
        }
        uBit.display.print(image);
    }


int main()
{
   uBit.init();
   uBit.io.P1.setDigitalValue(0);
   uBit.io.P2.setDigitalValue(1);

   MicroBitSerial serial(USBTX, USBRX);
   while (1) {
       int u = 3300 * uBit.io.P0.getAnalogValue()/ 1023;
       uBit.serial.printf("%d\r\n", u);
       ledDispay (u);
       uBit.sleep(500);
       }
}
```

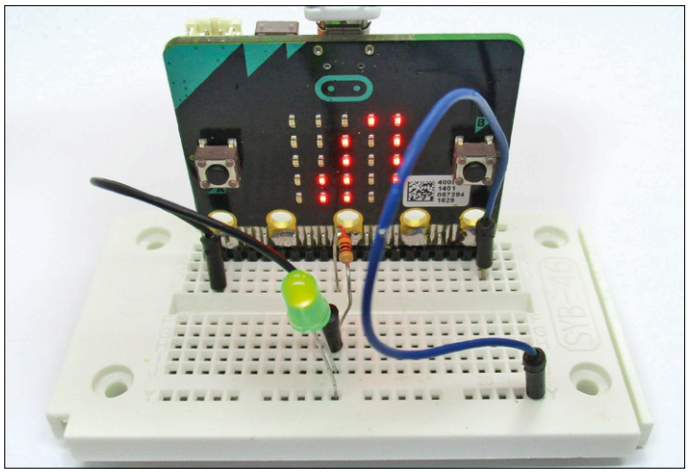


Bild 8. Messung der LED-Spannung 2,496 V.

bende Zahl wird mit der Integer-Variablen n übergeben. Im Display wird eine Spannung bis 3300 mV angezeigt (**Listing 3**).

Im Hauptprogramm werden zusätzlich zwei Portpins geschaltet. P1 wird low, P2 wird high. Eine Umschaltung der Datenrichtung muss man nicht explizit angeben, denn die passiert automatisch im Hintergrund. Man kann nun leicht die Belastbarkeit der Ports untersuchen und dazu gleich den eigenen AD-Wandler nutzen. Es zeigt sich, dass sie deutlich hochohmiger sind als die Ports eines AVR-Controllers. Offiziell sind die Ports in beiden Richtungen für bis zu 5 mA ausgelegt. Die Messungen zeigen, dass dann ein Spannungsabfall von 300 mV auftritt. Der On-Widerstand der Port-FETs beträgt also etwa 60 Ω. Messungen mit großen Lasten zeigen, dass der Ausgangsstrom niemals über 15 mA steigt. Beim direkten Anschluss einer LED ohne Vorwiderstand fließt ein Strom von etwa 10 mA.
Im Normalfall wird man aber einen Vorwiderstand verwenden. Und dann kann man gleich die LED-Spannung messen, verschiedene LEDs vergleichen und weitere Experimente durchführen (**Bild 8**). ⏮

(160273)

Weblinks

[1] http://microbit.org/code/

[2] https://developer.mbed.org/

[3] https://lancaster-university.github.io/microbit-docs/ubit/

[4] https://developer.mbed.org/handbook/Windows-serial-configuration

[5] B. Kainka, Micro:bit Praktikum, CreateSpace 2016

[6] www.elektormagazine.de/150652

[7] www.elektormagazine.de/160274

[8] www.elektormagazine.de/160273

Anzeige

Die neuesten Produkte
für Ihre neuesten Designs™

MOUSER ELECTRONICS

Mehr neue Produkte auf Lager als jeder andere Distributor.

Jetzt bestellen auf
mouser.de

Tipps und Tricks

Von Lesern für Leser

Hier kommen wieder clevere Lösungen, die das Elektronikerleben leichter machen.

0,5-mm-Pins löten

Von Peter Krengel

Als ich das Elektor Dezember-Heft durchblätterte, stieß ich auf den Artikel „0,5-mm-Beinchen löten“ und war sehr gespannt, welcher neue Trick mir nun präsentiert werden würde. Meine Neugierde wich bald einem Schmunzeln, denn diese 144-beinigen Biester kenne ich nur zu gut, weil ich viel mit FPGAs von Altera und Xilinx arbeite. Die Lösung, mit dem Kratzmesser an das Problem heranzugehen, schockte mich dann aber wirklich, denn es geht doch alles viel einfacher, sauberer und schneller!

Eigentlich sollte man in einem SMD-Ofen alle anderen Bauteile löten, nur nicht diese Tausendfüßler. Doch meine wirklich einfache Methode funktioniert auch dann, wenn durch falsches Löten im Ofen Kügelchen und Spratzer die Platine verunreinigen. Die Lösung des Problems heißt „Gull-Wing“-Lötspitze, wie man sie von Weller oder Ersa erwerben kann [1][2]. Dabei handelt es sich um eine Hohlspitze, in die ein Tröpfchen Zinn eingefüllt werden kann. Ich benutze eine solche Spitze mit der nachfolgend beschriebenen Technik. Vorab, Kratzen ist völlig unnötig, denn zu Kügelchen (typisch auch bei einem schlechten oder schlecht eingestellten SMD-Ofen) kommt es erst gar nicht. Und damit lassen sich auch 0,3 mm schmale Beinchen sehr sauber und ohne Kurzschlüsse löten. So geht es:

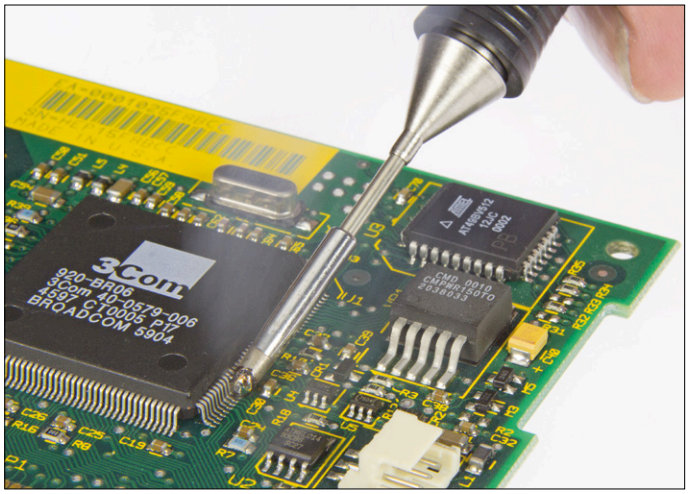
1. *Die Lötpads mit „No-Clean“-Flussmittel abreiben, alle Beinchen des ICs mit diesem Flussmittel benetzen. Das geht am besten mit einem Flussmittelstift, den man vorher irgendwo aufdrückt, damit aus der Spitze etwas Mittel herausquillt. Nicht zu gewalttätig vorgehen! Man darf die Beinchen dabei nicht verbiegen, sondern nur durch den an der Spitze hängenden Tropfen ziehen und dann trocknen lassen.*
2. *Den Tausendfüßler mit einer Lupe (zehnfache Vergrößerung) so exakt wie möglich auf die Pads platzieren und an einem Eck-Bein mit einer 0,2er-Spitze (zum Beispiel Weller RT) anlöten.*
3. *Die Lage des ICs noch einmal mit der Lupe kontrollieren und gegebenenfalls leicht korrigieren.*
4. *Die 0,2er- durch die Gull-Wing-Lötspitze ersetzen. Die Temperatur so einstellen, dass das Zinn gut fließt, aber nicht oxidiert. Diese Temperatureinstellung sollte man zuvor genau ausprobieren.*
5. *Dann füllt man die Gull-Wing-Spitze mit Zinn, setzt sie sofort auf ein Beinchen an der Ecke auf und zieht sie langsam über die Pins. Dies macht man mit allen vier Seiten.*
6. *Zum Schluss werden die Lötungen kontrolliert und bei Bedarf noch einmal mit der leeren Spitze abgezogen.*

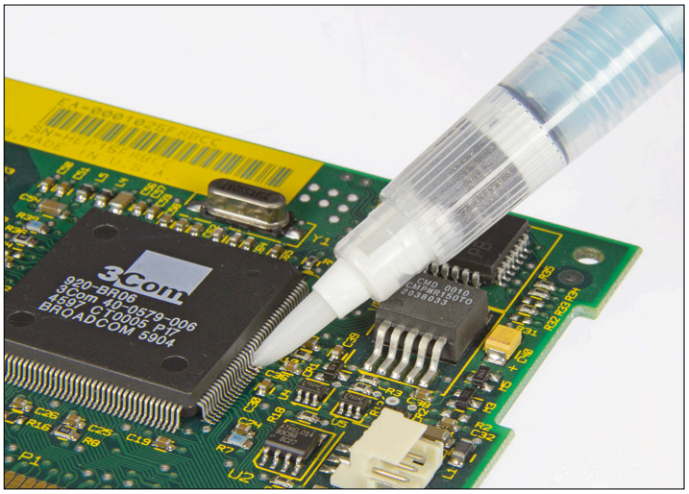
Und das Beste ist: Die 144 Beinchen sind in maximal zwei Minuten (inklusive Kontrolle) erschlagen. Eine Überhitzung ist durch die kurze Lötdauer ausgeschlossen. Probiert es mal aus - ihr werdet begeistert sein!

Die Lötstellen sehen aus wie maschinell gelötet, glänzen aber im Gegensatz zum maschinellen Löten richtig schön. Dies rührt wohl daher, dass die Lötzeit pro Pin durch das Drüberziehen sehr kurz ist und das Zinn keine Zeit hat, zu oxidieren, auch nicht bei leichter Übertemperatur. Zudem sorgt die Oberflächenspannung des nicht oxidierten Zinns für eine sehr gleichmäßige Zinnverteilung, denn Flüssigkeiten möchten ja am liebsten von der Masse her gleichmäßig strukturierte Körper bilden, etwa wie Wasser in der Schwerelosigkeit (aber wem sage ich das ;)). Auf jeden Fall findet sich das Zinn schlussendlich gleichmäßig verteilt und nur dort, wo es hingehört, nämlich an und auf den Pads.

[1] Weller: www.weller.de/en/Weller--Products--Product-details.html?article_id=D0719205001379336391A111723

[2] Ersa Micro: www.ersa-shop.com/serie-f%C3%BCr-micro-tool-hohlkehle-163mm-c-80_81_125_130.html





Rettung versauter IC-Lötstellen

Von Peter Krengel

Auch missratene IC-Lötstellen lassen sich mit einer Gull-Wing-Spitze retten. Dazu benötigt man wieder einen No-Clean-Flussmittel-Filz/Tropfstift.

1. *Reichlich No-Clean-Flussmittel rundherum auf den Beinchen des eingelöteten ICs so verteilen, dass es auch ein wenig unter den Chip läuft und*
2. *sofort nach Verlaufen des Flussmittels (es trocknet sehr schnell!) die Beinchen an allen Seiten des ICs mit der leeren Gull-Wing-Spitze abziehen.*
3. *fertig!*

Statt des No-Clean-Flussmittels kann man auch deutlich billigeres Kolophonium verwenden. Man legt sich ein (abgebranntes) Streichholz, eine gute, saubere Zahnbürste und ein 100-ml-Fläschchen 99-%igen Ethanol zurecht (nicht trinken, nur zurechtlegen!). Eine solch kleine Menge Ethanol kann man zu Apothekerpreisen in der Apotheke erwerben, preisgünstiger ist er im Netz. Keinen Spiritus oder etwa Isopropanol verwenden, denn beides hätte hässliche weiße Flecken auf der Platine zur Folge, die nur schwer zu entfernen sind.

Das Kolophonium wird in der Blechdose verflüssigt, dann nimmt man mit dem Streichholz ein oder mehrere kleine Tröpfchen auf und verteilt sie auf den Beinchen des ICs. Auch hier muss schnell gearbeitet werden. Falls das Kolophonium zu schnell fest wird, kann man die Platine mit einem Fön oder besser mit einer Heißluftpistole bei etwa 120...130 °C vorwärmen.

Wie gehabt zieht man die Beinchen dann mit der leeren Gull-Wing-Spitze ab, wenn nötig auch mehrfach. Alle Lötzinn-Spratzer sollten sich nun auf den Pads beziehungsweise den Beinchen gesammelt haben und somit saubere Lötungen ergeben.

Zahnbürste und Alkohol bleiben nicht ungenutzt, denn wenn mit Kolophonium gearbeitet wird, ist im Gegensatz zur No-Clean-Methode eine Reinigung mit Ethanol unumgänglich. In der Regel schwimmen nämlich die Beinchen nach dem Abziehen noch im Flussmittel. Bevor man sich an die Reinigung macht, muss die Platine aber unbedingt abgekühlt sein! Zur Not darf man sie auch fünf Minuten in die Tiefkühltruhe legen.

Das überschüssige Kolophonium wird wieder mit dem Streichholz (das vermeidet Kratzer auf der Platine) durch leichtes Schaben zerbröselt. Nach dem Entfernen des Kolophoniumstaubs (Pusten, Klopfen) werden alle Reste mit viel Alkohol und der Zahnbürste abgeschrubbt - fertig!

Man kann auch Sau-Pech haben!

Von Peter Krengel

Hier ein kleiner Tipp für alle, die es leid sind, zu viel Geld für zu wenig Kolophonium zu bezahlen. Im Netz sucht man nach Brühpech, Brühharz oder eben Saupech. Dabei handelt es sich um reines Kolophonium, das im Metzgereibedarf für kleines Geld und in rauen Mengen in Pulverform zu haben ist. Es dient dort der Entfernung von Federkielen bei Geflügel und Haarresten bei Schweinen. Das Harz kann man durch Erhitzen verflüssigen und dann in fester Form in einer kleinen Cremedose, die man zuvor mit Alkohol und Spülmittel gut gereinigt hat, aufbewahren.

Entlötlitze reinigen

Von Peter Krengel

Kolophonium ist auch nützlich bei „stumpf" gewordener Entlötlitze. Man legt die Entlötlitze in verdünnte, 5...10-%ige Salzsäure aus dem Baumarkt und entfernt damit das Kupferoxid, das sich in der Salzsäure löst. Danach die Litze gut mit Wasser abspülen und sodann durch verflüssigtes Kolophonium ziehen, dieses hart werden lassen und abschließend überschüssiges Harz abbröckeln. So gut hat Ihre Litze noch nie gesaugt, wetten dass? ◀

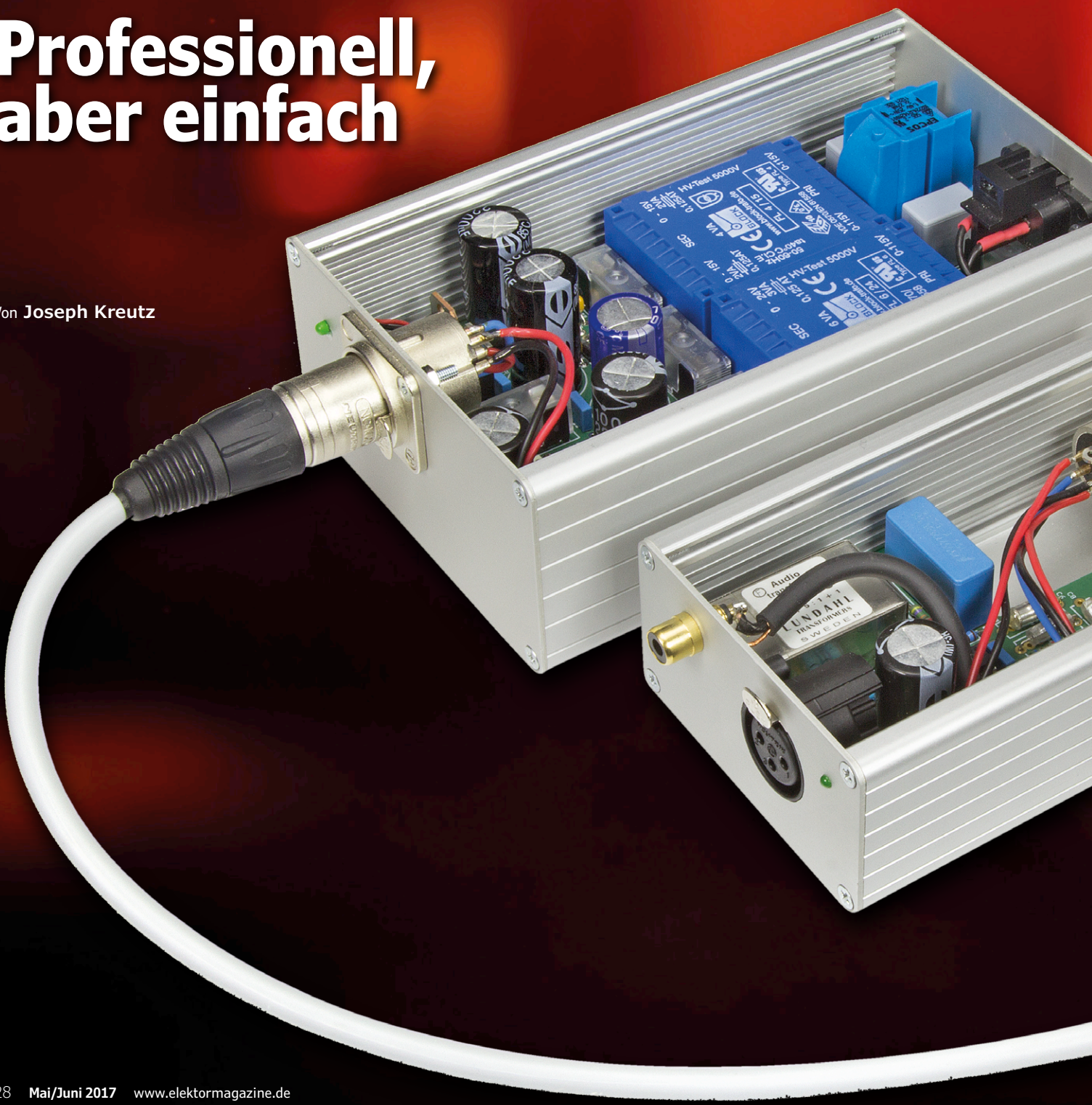
(160324)

Sie haben selbst eine clevere Lösung für etwas wirklich Fummeliges? Wenden ein Bauteil oder Werkzeug auf ungewöhnliche Weise an? Haben eine Idee, wie man ein Problem einfacher oder besser angehen könnte, als das bisher gelöst wurde? Schreiben Sie uns – für jeden Tipp, den wir veröffentlichen, loben wir 40 Euro aus!

Mikrofon-Vorverstärker

Professionell, aber einfach

Von **Joseph Kreutz**



Mit einem Kassettenrekorder nimmt niemand mehr auf – heute kommt oft ein Notebook zum Einsatz. Doch während die Line-Eingänge vieler PCs zumeist eine gute Klangqualität aufweisen, befinden sich die Mikrofon-Eingänge vieler Notebooks nur auf einem mittelmäßigen Niveau, das durch Rauschen und Verzerrungen gekennzeichnet ist. Außerdem lässt sich in der Regel nur ein Mikrofon anschließen - von professionellen Mikrofonen mit symmetrischen Anschlüssen ganz zu schweigen, die außerdem oft noch eine Phantomspeisung von 48 V benötigen. Hier kommt ein Vorverstärker, der diese Nachteile nicht aufweist und vor allem die Besitzer von High-End-Soundkarten begeistern wird.

Verstärkerschaltung

Bild 1 zeigt den Schaltplan des Vorverstärkers. Der auf den ersten Blick etwas veraltet erscheinende Trafo (TR1) dient zur galvanischen Trennung des Eingangs und zur Gleichtaktunterdrückung (common mode rejection). Der mit einem Übertragungsverhältnis von 1:10 gut erhältliche Trafo unterdrückt außerdem spürbar die durch Mobiltelefone verursachten Störungen. Außerdem wird die Signalspannung durch ihn um 20 dB verstärkt, was auch dem Signal-Rausch-Verhältnis zugutekommt. Um die gewonnenen Vorteile nicht durch die nachfolgenden Stufen wieder zu verlieren, wurde ein Operationsverstärker mit FET-Eingängen verwendet, dessen Rauschen vernachlässigbar ist. Ein AD743 wäre für die Schaltung aufgrund seines niedrigen Eingangsrauschens sehr gut geeignet (2,9 nV/√Hz [1]), ist aber teuer. Mit der wesentlich preisgünstigeren Variante OPA134 von Burr-Brown (TI) haben wir eine gute Alternative gefunden. Allerdings hatte der Autor für den Prototypen des Vorverstärkers einen TL071 eingesetzt, der ebenfalls ein erstaunlich gutes Resultat erzielte [2]. Die Doppel-Opamps wie der OP2134 oder der TL072 eignen sich dann natürlich recht gut für eine entsprechende Stereo-Variante. Bei der Suche nach einem geeigneten Trafo lohnt sich ein Blick auf die Seiten von Bürklin [3], der zahlreiche Exemplare mit einem guten Preis-Leistungs-Verhältnis anbietet. Entsprechende Trafos gibt es übrigens auch von Lundahl [4], Sowter [5] und Jensen [6], aber die betreffenden Preise bewegen sich hier eher in höheren Regionen, so dass sich eine gezielte Suche bei verschiedenen Anbie-

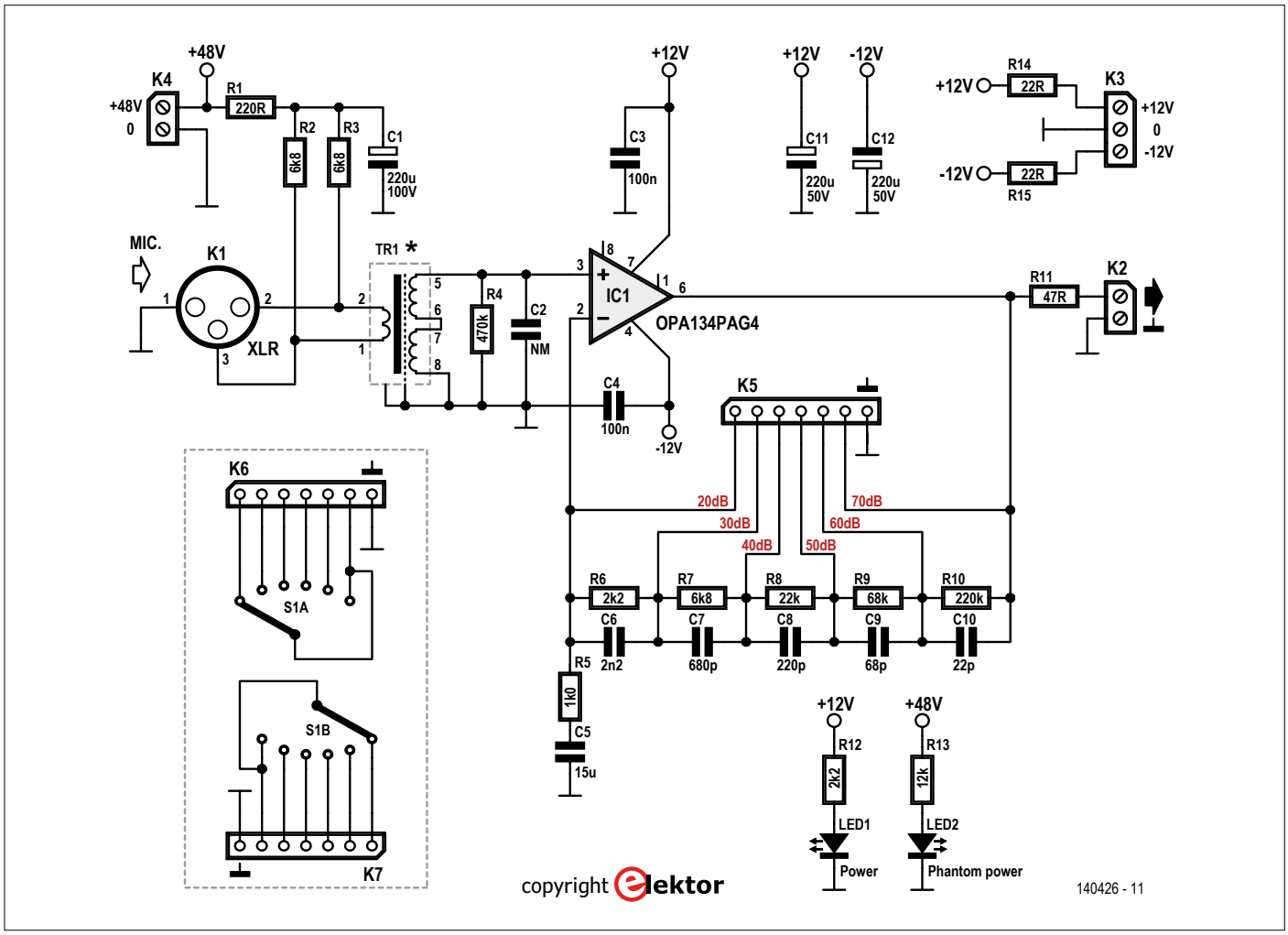


Bild 1. Der Operationsverstärker ist sozusagen das Herz der Schaltung.

tern durchaus lohnen kann. Bedenken Sie, dass für den Trafo TR1 auch andere Modelle in Frage kommen können, wie zum Beispiel der ÜP3095M und ÜP3096M von Pikatron, LL1935 von Lundahl und JT-115K-EPC von Jensen. Der Trafo LL1935 von Lundahl besitzt zwei primäre und zwei sekundäre Wicklungen. Hierzu eine Anmerkung aus dem Labor: Bei manchen Trafos anderer Hersteller kann es (bei derselben Pin-Konfiguration) vorkommen, dass eine Wicklung in Bezug zur anderen umgekehrt gepolt ist. In diesem Falle bitte Vorsicht: Kurzschlussgefahr! Doch zurück zur Schaltung: Die Verbindung der Primärspule mit dem verwendetem Mikrofon kann über eine professionelle XLR-Buchse oder über eine 6,35-mm-Stereo-Klinkenbuchse erfolgen. Im letzteren Falle entspricht die äußerste Spitze des Steckers dem „heißen" Ende (linker Kanal), der mittlere Steckerteil dem „kalten" Ende (rechter Kanal), und der Rest ist die Masse.

Kondensator-Mikrofone benötigen eine Versorgungsspannung von 48 V, die über die Widerstände R2 und R3 (jeweils 6,8 k) eingespeist wird. Die absoluten Widerstandswerte müssen nicht genau eingehalten werden, aber sie dürfen gegenseitig nur um 0,4 % (oder weniger) voneinander abweichen [7][8], so dass man am besten eine Auswahl mit einem Ohmmeter vornimmt. Damit sie bei einem Kurzschluss gegen Masse nicht durchschmoren, sollten die Widerstände eine Belastbarkeit von mindestens 5 W aufweisen. Ein Sekundäranschluss des Trafos (TR1) ist mit Masse verbunden, während der andere direkt zum positiven Eingang des Opamps (IC1) führt. Der negative Eingang des Verstärkers ist mit einem Gegenkopplungsnetzwerk verbunden, das den Verstärkungsfaktor festlegt. Der Mittelkontakt (Schleifer) des sechsfachen Drehschalters sorgt dafür, dass der Ausgang des Opamps entweder direkt oder über die seriellen Widerstände R6 bis R10 mit seinem negativen Eingang verbunden wird. Unter Berücksichtigung der bereits durch den Trafo erfolgten Verstärkung kann die Verstärkung des Vorverstärkers zwischen 20 und 70 dB variieren.

Beim Schalter sollte es sich um ein „Make before break"-Exemplar handeln: Der neue Kontakt wird geschlossen, bevor der alte unterbrochen wird. Auf diese Weise werden abrupte Sprünge der Verstärkungsfaktoren vermieden und durch einen sanften Übergang ersetzt, der höchstens von einer akustischen Rückkopplung begleitet sein kann.

Die beiden zu R6 und R10 parallelen Kondensatoren begrenzen den Frequenzgang des Vorverstärkers auf 32 kHz. Im Prototyp wurden dazu axiale Polystrol-Versionen höchster Qualität verwendet – es sind allerdings auch Kondensatoren mit Keramik oder Kunststoff erlaubt. An die Sekundärwicklung von Tr1 muss gegebenenfalls noch ein Kondensator (C2) angeschlossen werden, um unerwünschte Resonanzen des Trafos zu unterdrücken. Er kann jedoch in den meisten Fällen weggelassen werden.

Noch ein paar Tipps: Die Entkopplungskondensatoren C3 und C4 von 100 nF sollten möglichst nahe am Operationsverstärker montiert werden. Bei den Widerständen sollte es sich um Metallfilm-Exemplare mit einer Toleranz von 1 % handeln – vor allem bei R5 bis R10. R4 (470 k) kann entfallen, wenn für das Mikro keine spezielle Eingangsimpedanz angegeben ist. Andernfalls muss ein Wert eingesetzt werden, der dem Hundertfachen der empfohlenen Impedanz entspricht.

Die Elkos müssen eine Spannung von 25 V vertragen, bis auf C1 (63 oder 80 V). Die Versorgungsspannung des Verstärkers ist relativ unkritisch, sofern sie in dem vom Hersteller angegebenen Bereich liegt. Wenn ausschließlich Mikrofone ohne zusätzliche 48-V-Versorgung betrieben werden, können C1 und R1 bis R3 entfallen. C5 hat einen Wert von 15 µF, aber wenn noch tiefere Frequenzen im Übertragungsweg abgeschnitten werden sollen (oder zumindest die Phase im tiefen Bereich verschoben werden soll), kann auch ein Kondensator gleicher Bauart von 22 µF eingesetzt werden.

INFOS ZUM PROJEKT

Audio
Vorverstärker
Mikrofon

➡ Einsteiger
Fortgeschrittener
Experte

Etwa 5 Stunden (ohne Gehäuse)

Nichts Spezielles

Rund 270 € (mit Gehäuse)

Versorgung ±12 V

Der Operationsverstärker (IC1) benötigt eine bipolare Spannungsversorgung von jeweils +12 V und −12 V, die durch die im unteren Bereich der Schaltung in **Bild 3** gezeigten Bauteile realisiert wird – aufgrund der beiden Stabilisatoren eine extrem klassische Art von Spannungsquelle. Es ist empfehlenswert, möglichst nahe an den Stabilisatoren jeweils auch Entkopplungskondensatoren von 100 nF anzubringen. Die Gleichrichtung wird durch eine Brücke aus vier Dioden des Typs 1N4007

Bild 2. Platinenlayout des Verstärkers.

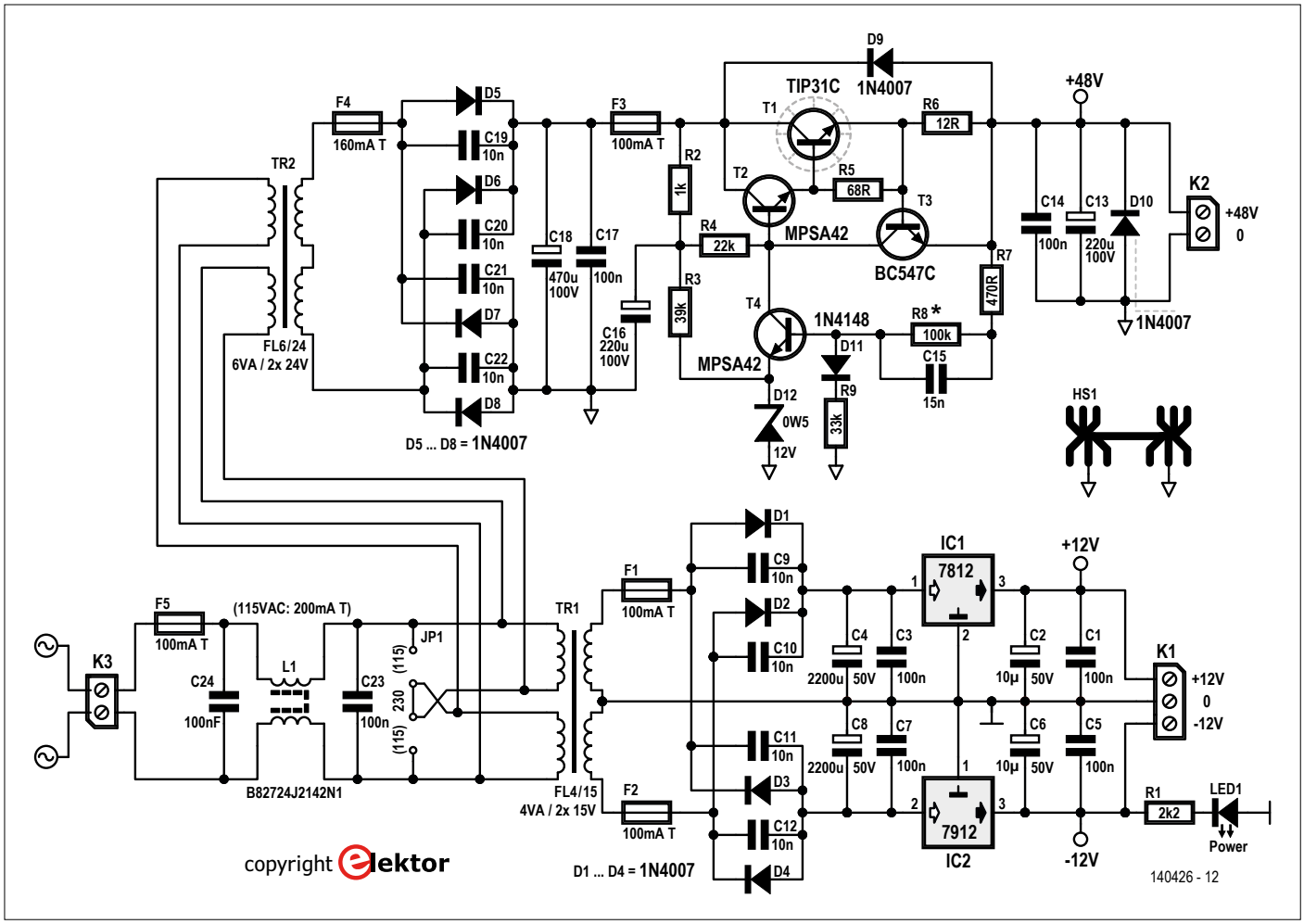


Bild 3. Schaltbild des doppelten Netzteils: ±12 V und 48 V.

realisiert. Die Sicherung F5 zum Schutz des Trafos wurde nach den Angaben des Autors bemessen.

Die Leuchtdiode (LED1) auf der Frontplatte zeigt an, ob der Verstärker in Betrieb ist oder nicht. Bestimmte Exemplare des Stabilisators 7912 liefern ihre nominale Spannung erst dann, wenn mindestens ein Strom von 5 mA fließt. Wenn man auf die LED verzichten möchte, sollte der Ausgang des Stabilisators über einen Widerstand von 2,2 k mit Masse verbunden werden, damit ein ausreichender Strom gewährleistet ist. Der Operationsverstärker benötigt nämlich maximal nur einen Strom von 5 mA bei 25 Grad Umgebungstemperatur. Aus diesem Grunde benötigen die „Stabis" auch keine Kühlkörper – auch nicht bei Stereo-Betrieb.

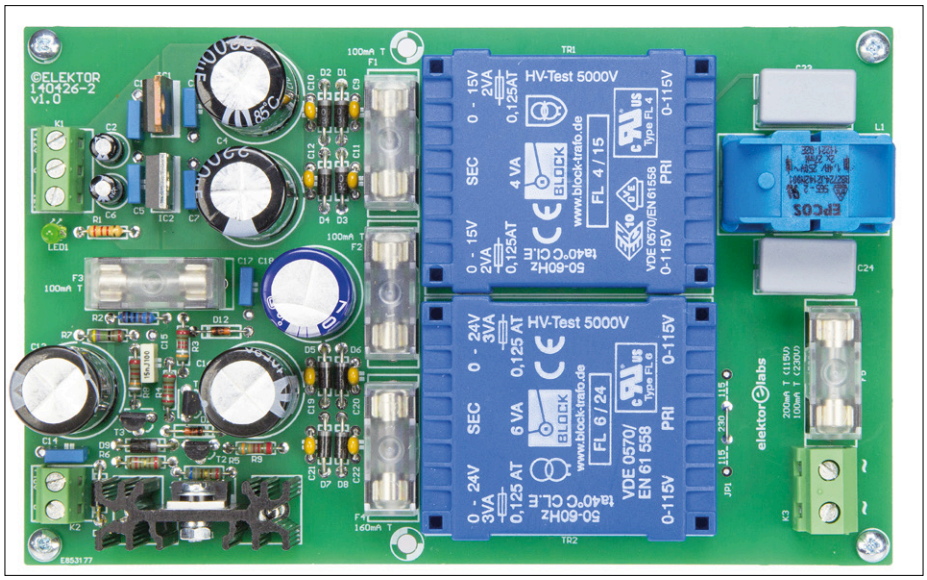


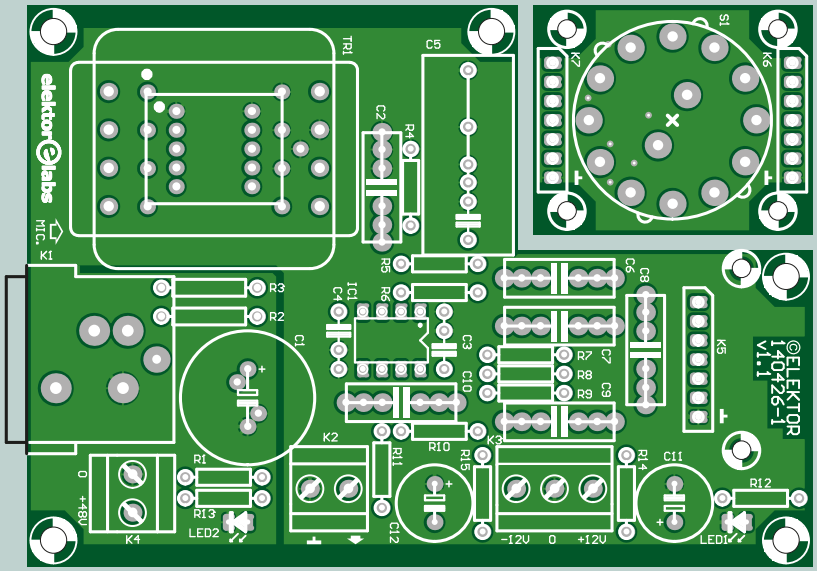
Bild 4. Platinenlayout des doppelten Netzteils.

Auch die Netzteile von Desktoprechnern erzeugen die geforderten, positiven und negativen Spannungen von 12 V. In diesem Falle sollten die betreffenden Spannungen jedoch sorgfältig gefiltert werden, um die durch das Motherboard und andere Einheiten erzeugten Störungen zu unterdrücken. Außerdem sollte man aus Sicherheitsgründen in jeden der beiden Versorgungszweige eine Sicherung von 100 mA einsetzen, damit auch der Rechner im Falle einer Störung keinen Schaden erleidet.

Versorgung 48 V

Im oberen Teil von Bild 3 ist das Netzteil für die 48-V-Versorgung eingezeichnet.

STÜCKLISTE – VORVERSTÄRKER



Widerstände:

Alle Widerstände Metallfilm
R1 = 220 Ω, 1%, 250 mW
R2, R3 = 6,8 k, 0,1%, 500 mW
R4 = 470 k, 1%, 400 mW
R5 = 1 k, 1%, 600 mW
R6,R12 = 2,2 k, 1%, 400 mW
R7 = 6,8 k, 1%, 400 mW
R8 = 22 k, 1%, 250 mW
R9 = 68 k, 1%, 250 mW
R10 = 220 k, 1%, 250 mW
R11 = 47 Ω, 1%, 250 mW
R13 = 12 k, 1%, 250 mW
R14,R15 = 22 Ω, 1%, 250 mW

Kondensatoren:

C1 = 220 µF, 100 V, 20%, in Rastermaßen von 5 oder 7,5 mm, $Ø_{max}$ 17,8 mm
C2 = nicht montiert
C3,C4 = 100 n, 100 V, 5%, in Rastermaßen von 5 oder 7,5 mm, PET
C5 = 15 µ, 63 V, 10%, in Rastermaßen von 5/7,5/10/15/22,5 mm, PET
C6 = 2,2 n, 1%, 160 V, in Rastermaßen von 5/7,5/10/14,6 mm, Polystrol, axial
C7 = 680 p, 2,5%, 160 V, in Rastermaßen von 5/7,5/10/14,6 mm, Polystrol, axial
C8 = 220 p, 2,5%, 160 V, in Rastermaßen von 5/7,5/10/14,6 mm, Polystrol, axial
C9 = 68 p, 2,5%, 160 V, in Rastermaßen von 5/7,5/10/14,6 mm, Polystrol, axial
C10 = 22 p, ±1 p, 160 V, in Rastermaßen von 5/7,5/10/14,6 mm, Polystrol, axial
C11,C12 = 220 µ, 50 V, 20%, Rastermaß 5 mm, $Ø_{max}$ 10 mm

Induktivitäten:

TR1 = Transformator LL1935, 1:10 (1:1:5+5), Lundahl, 38 × 23 × 16 mm

Halbleiter:

LED1 = LED, grün, 3 mm, T-1
LED2 = LED, blau, 3 mm, high intensity, T-1
IC1 = OPA134PAG4, DIP-8

Außerdem:

K1 = XLR-Audio-Buchse, 3-polig, female, Montage auf Platine (Neutrik NC3FAAH1)
K2,K4 = 2-polige Klemmleiste, Rastermaß 5,08 mm, 630 V
K3 = 2-polige Klemmleiste, Rastermaß 5,08 mm, 630 V
K5 = 7-polige Stiftleiste, male, Rastermaß 2,54 mm, rechtwinklig
K6,K7 = nicht montiert (Rückseite von K5)
S1 = 6-fach Drehschalter, 2 Ebenen, MBB-Funktionalität, Lorlin CK1060
Platine 140426-1
Gehäuse 1455K1201, Hammond Manufacturing

Weitere Trafos auf Platine:

TR1 = Pikatron ÜP 3095 M, 1:10, 18×14,5×13,2 mm (siehe [3])
TR1 = Pikatron ÜP 3096 M, 1:15, 18×14,5×13,2 mm (siehe [3])
TR1 = Jensen transformers JT-115K-EPC, 1:10, 31,75×31,75×23,62 mm (siehe [6])

Die Norm EN 61938 [8] sieht für Mikrofone einen Höchststrom von 10 mA vor und erlaubt eine Abweichung von +/- 4 V um den nominalen Wert. Die Netzspannung wird mittels TR2 in eine Spannung von 2 x 24 V verwandelt. Die konventionelle Stabilisierung besteht hier nur aus diskreten Bauteilen. Die 12-V-Zenerdiode liefert eine entsprechende Referenzspannung. Zur Spannungsregelung wird das gegenkoppelnde Signal über R7 und R8 an Transistor T4 geleitet. T1, ein TIP31C, ist großzügig dimensioniert, um gegen eventuelle Überlastungen gewappnet zu sein. Er befindet sich dazu noch auf einem Kühlkörper, da er sich durch die höhere Eingangsspannung erwärmt. Ein thermischer Widerstand von etwa 20 °C/W dürfte zur Kühlung von T1 ausreichen. In Notfällen verringert immer noch die Sicherung F3 von 100 mA einen eventuellen Schaden. T3 und R6 begrenzen den durch dieses Netzteil erzeugten Strom auf 50 mA – mehr als genug für die meisten Mikros. Die Elkos des 48-V-Netzteils müssen eine Arbeitsspannung von 100 V aushalten. Das Elektor-Labor hat eine Restwelligkeit kleiner als 0,2 mVss bei 10 mA gemessen, die auf der Verstärkerplatine weiter unterdrückt wird.

Zusammenbau

Die Primäranschlüsse der Trafos TR1 und TR2 sind sowohl für 230 als auch für 115 V geeignet: Bei 230 V ist eine Verbindung für JP1 erforderlich, bei 115 V benötigt man zwei davon. Achtung: Niemals drei Verbindungen einsetzen. Das ist auf der Platine auch deutlich markiert. Bei 115 V muss der Wert der Sicherung verdoppelt werden. Laut Trafohersteller sollten Sicherungen nur auf der Sekundärseite angebracht werden. Allerdings haben wir aus Sicherheitsgründen auch auf der Primärseite Sicherungen vorgesehen. Um die Platinen auch in einem schönen Gehäuse unterzubringen, hat das Elektor-Labor zwei Versionen aus der Serie 1455 von Hammond ausgewählt (siehe Foto am Anfang des Artikels). Der Verstärker befindet sich im

kleinen Gehäuse. Achtung: Das Metallgehäuse des Jensen-Trafos kann unter Umständen die Leiterbahnen auf der Platine kurzschließen. Beim Einlöten dieses Bauteils daher bitte genug Abstand zur Platine einhalten oder eine etwa 0,5 mm dicke Isolationsschicht verwenden.

Sicherheitshinweis

Das Netzkabel ist direkt an der Platine angeschlossen - und zum Gehäuse besteht überhaupt keine Verbindung. Dies garantiert eine Isolierung der Klasse II. Alle Leitungen, Kontakte und Leiterbahnen, die direkt mit der Netzspannung verbunden sind, müssen sich mindestens 6 mm von irgendwelchen Metallteilen des Gehäuses entfernt befinden. Schneiden Sie auch bitte die auf der Leiterbahnseite herausragenden Enden der Netzspannungsleitungen so kurz wie möglich ab. Für das Netzkabel muss ein flexibles

STÜCKLISTE - NETZTEIL

Widerstände:

R1 = 2,2 k, 5%, 0,25 W, Kohleschicht
R2 = 1,0 k, 1%, 0,6 W, Metallfilm
R3 = 39 k, 1%, 0,25 W, Metallfilm
R4 = 22 k, 1%, 0,25 W, Metallfilm
R5 = 68 Ω, 1%, 0,4 W, Metallfilm
R6 = 12 Ω, 1%, 0,25 W, Metallfilm
R7 = 470 Ω, 1%, 0,25 W, Metallfilm
R8 = 100 k, 1%, 0,25 W
R9 = 33 k, 1%, 0,25 W

Kondensatoren:

C1,C3,C5,C7,C14,C17 = 100 n, 100 V, 5%, Rastermaß 5/7,5 mm, PET
C2,C6 = 10 µ, 50 V, 20%, Rastermaß 2 mm, Ømax 6,5 mm.
C4,C8 = 2200 µ, 50 V, 20%, in Rastermaßen von 5 oder 7,5 mm, 17,8 mm max.
C9,C10,C11,C12,C19,C20,C21,C22 = 10 n, 200 V, 10%, Rastermaß 5,08 mm, keramisch X7R
C13,C16 = 220 µ, 100 V, 20%, in Rastermaßen von 5 bis 7,5 mm, 17,8 mm max.
C15 = 15 n, 100 V, 5%, in Rastermaßen von 5 bis 7,5 mm
C18 = 470 µ, 100 V, 20%, in Rastermaßen von 5 bis 7,5 mm, 17,8 mm max.
C23,C24 = 100 n, X1, 440 V AC, in Rastermaßen von 10/12,5/15 mm, Polypropylen

Induktivitäten:

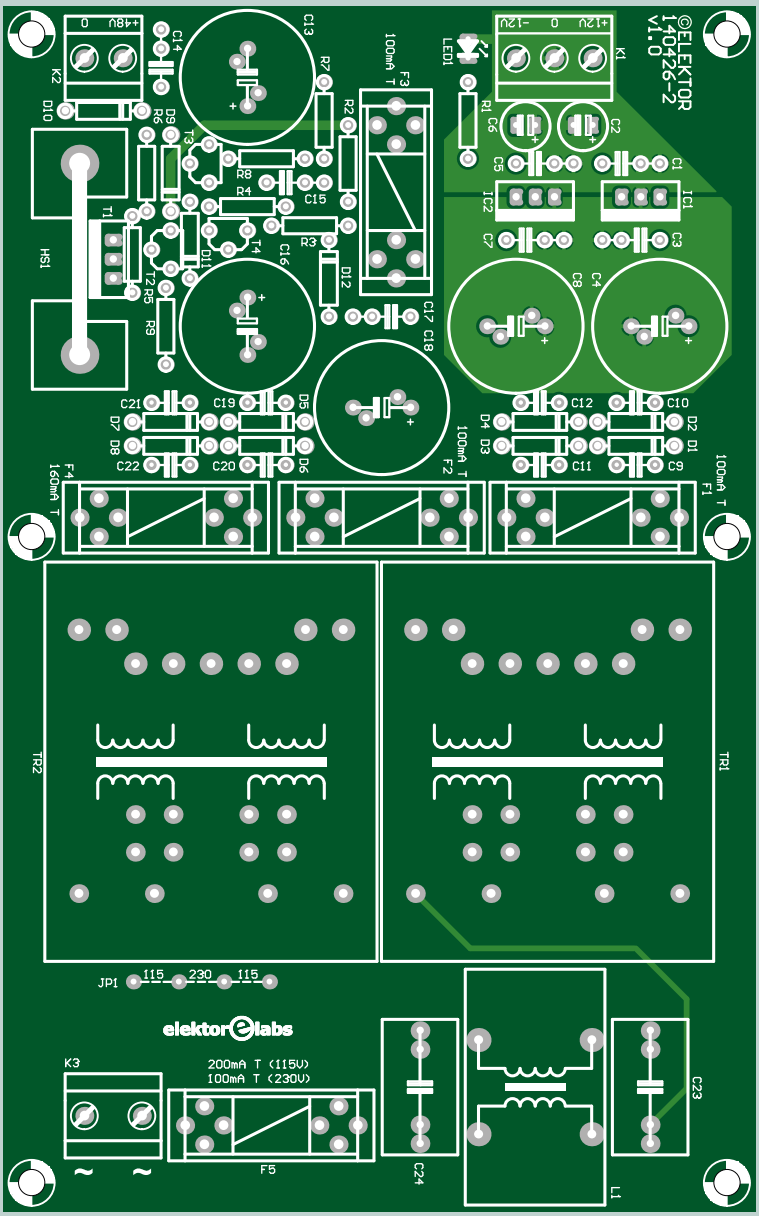
L1 = B82724J2142N1, Drossel, common mode, 2×27 mH, 1,4 A
TR1 = FL4/15 Block, prim. 2×115 V, sec. 2×15 V, 2 VA, Montage auf Platine
TR2 = FL6/24 Block, prim.2×115 V, sec. 2×24 V, 6 VA, Montage auf Platine

Halbleiter:

D1,D2,D3,D4,D5,D6,D7,D8,D9,D10 = 1N4007, 1000 V, 1 A, DO-41
D11 = 1N4148, 100 V, 200 mA, DO-204AH
D12 = BZX55C12, Zenerdiode 12 V/0,5 W, DO-35
LED1 = LED, grün, 3 mm, T-1
T1 = TIP31C, NPN, TO220 (100 V/40 W/3 A)
T2,T4 = MPSA42, NPN, TO-92 (300 V/625 mW/500 mA)
T3 = BC547C, NPN, TO-92 (45 V/500 mW/100 mA)
IC1 = 7812, TO-220-3
IC2 = 7912, TO-220AB-3

Außerdem:

JP1 = Kabel 0,6 mm (siehe Text)
K1 = 3-polige Klemmleiste, Rastermaß 5,08 mm, 630 V
K2 = 2-polige Klemmleiste, Rastermaß 5,08 mm, 630 V
K3 = 2-polige Klemmleiste, 7,62 mm, 630 V
F1,F2,F3,F5 (*1) = Sicherung, 100 mA, träge, 250 V, 20×5 mm



F4 = Sicherung, 160 mA, träge, 250 V, 20×5 mm
F1,F2,F3,F4,F5 = Sicherungshalter, 20×5 mm, 500 V, 10 A
F1,F2,F3,F4,F5 = Abdeckung für Sicherungshalter 20×5 mm
HS1 = Kühlkörper SW25-4 Aavid Thermalloy, 13 °C/W, 34,5×12,5×25 mm
Platine 140426-2
*1 für 115 VAC: 200 mA, träge, 250 V, 5×20 mm
Gehäuse 1455N1601, Hammond Manufacturing

Kabel von 0,75 mm² verwendet werden. Nun muss nur noch das Kabel zwischen Netzteil und Verstärker vorbereitet werden. Es ist mit zwei fünfpoligen XLR-Steckern ausgerüstet. Das mag überdimensioniert erscheinen, aber diese Stecker und die dazugehörigen Buchsen lassen sich sehr gut und bequem montieren und verlöten. Natürlich können Sie auch andere, geeignete Stecker verwenden oder die Versorgung sogar auf zwei Kabel aufteilen. ⏮

(140426)

XLR-Stecker und Kabel für die Versorgung des Vorverstärkers.		
Versorgung	**Mikro-Vorverstärker**	**Kabel**
Geräte-Buchse XLR-Buchse (female), Neutrik NC5FD-LX, Montage auf Platine (Farnell 1390124)	**Geräte-Buchse** XLR-Buchse (male), Neutrik NC5MP, Montage auf Platine (Farnell 250820)	**Vorschlag Elektor-Labor:** Pro Power PPCY4C0.5 (Meterware), abgeschirmt, 4 Innenleiter von je 0,5 mm2, $\varnothing_{ext}$ 6,5 mm
Kabelstecker XLR-Stecker (male), Neutrik NC5MX, Montage am Kabel (Farnell 250764)	**Kabelstecker** XLR-Stecker (female), Neutrik NC5FXX, Montage am Kabel (Farnell 1108204)	**Alternativ:** Pro Power PPCY4C0.75 (Meterware), abgeschirmt, 4 Innenleiter von je 0,75 mm2, $\varnothing_{ext}$ 7 mm

Kurven aus dem Elektor-Labor (erstellt mit Präzisions-Audio-Analyzer)

Bedingungen in den beiden Fällen:

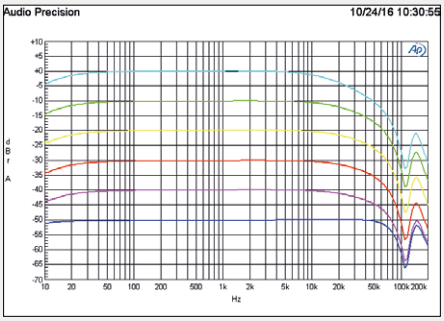
- Quellimpedanz = 200 Ω;
- Gain min. = 19,4 dB;
- Ausgangsspannung max. = 7,8 V (THD 0,1%, Gain 70 dB)

Farben der Kurven in Abhängigkeit der Gesamtverstärkung:
70 dB (hellblau), 60 dB (grün), 50 dB (gelb), 40 dB (rot), 30 dB (rosa) und 20 dB (dunkelblau).

Tabelle 1. Messung mit dem **Lundahl LL1935**

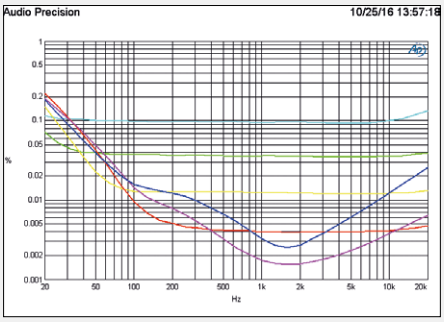
Gain	THD+N (Quellimpedanz 200 Ω) 1 kHz, 1 V_{Aus}, B 22 kHz	S/N (Quellimpedanz 200 Ω) in Bezug auf 1 V_{Aus}, B 22 kHz	Eckfrequenz bei – 3 dB tief/hoch
20	0,0032 %	110 dB / 112,7 dBA	< 10 Hz / 74 kHz
30	0,0014 %	100 dB / 103 dBA	12,2 Hz / 37 kHz
40	0,0028 %	90,8 dB / 93,1 dBA	12,8 Hz / 34 kHz
50	0,0087 %	80,8 dB / 83 dBA	12,8 Hz / 34 kHz
60	0,027 %	71 dB / 73 dBA	12,8 Hz / 30,6 kHz
70	0,078 %	62 dB / 63,5 dBA	12,8 Hz / 17 kHz

Amplitude als Funktion der Frequenz bei verschiedenen Verstärkungsfaktoren im Bereich von 10 bis 200 kHz.

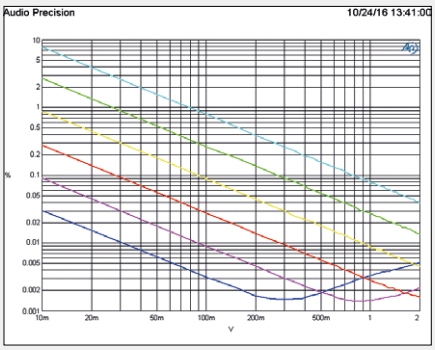


Bei 20 dB (dunkelblau) stimmt die Kurve praktisch mit der des Trafos überein. IC1 verhält sich wie ein Puffer. Bei 151 kHz gibt es eine Resonanzspitze, deren Amplitude und Frequenz weitgehend von der Impedanz und der Symmetrie der Quelle abhängen (symmetrisch oder asymmetrisch). Für die Verstärkungswerte 30, 40 und 50 sind die Eckfrequenzen fast gleich, aber bei 60 und 70 dB beginnt sich die Bandbreite des Opamps bemerkbar zu machen. Bei einer Verstärkung von 50 dB beträgt die Bandbreite des Opamps 8 MHz/316, also 25,3 kHz! Wer eine höhere Bandbreite benötigt, muss daher zu anderen Opamps greifen, zum Beispiel zum LME49710 (55 MHz) oder zum OPA627 (16 MHz). Der Austausch von C5 durch ein Exemplar 22 µF senkt die Eckfrequenz auf 9,6 kHz, und ändert die Verstärkung bei 20 Hz von -1,48 dB auf -0,87 dB.

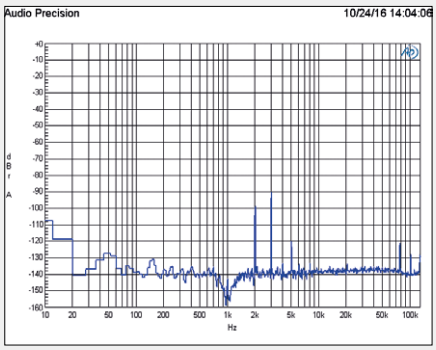
Verzerrung und Rauschen in Abhängigkeit der Frequenz bei 1 V Ausgangsspannung, verschiedenen Verstärkungsfaktoren und 80 kHz Bandbreite.



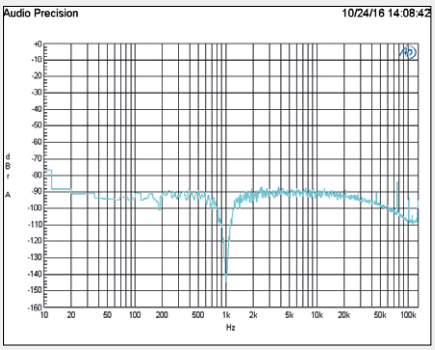
Da die Ausgangsspannung hier ungeachtet des Verstärkungsfaktors gleich bleiben soll, muss die Höhe des Eingangssignals entsprechend verändert werden. Für die höchste Verstärkung genügt ein Eingangspegel von 335 µV, um eine Spannung von 1 V am Ausgang zu erreichen. Bei höheren Pegeln wird vor allem das Rauschen größer, und bei geringeren Pegeln steigen die Verzerrungen an und werden zu einem wichtigen Faktor. Der Anstieg der Verzerrung an den Enden der Kurven (bei höheren Eingangsspannungen) ist grundsätzlich auf den Eingangstrafo zurückzuführen.



Verzerrung und Rauschen als Funktion des Ausgangspegels bei 1 kHz und einer Bandbreite von 22 kHz.
Verzerrungen an den Anstiegen der Extremwerte der Kurven bei den höchsten Verstärkungsgraden.



Fast Fourier Transformation (FFT) bei 1 kHz für die geringste Gesamtverstärkung (20 dB) und V_{Aus} = 1 V.
Eine dritte Harmonische mit -90 dB ist sichtbar, bedingt durch den Wert der „total harmonic distortion + noise“ von 0,0032 %.

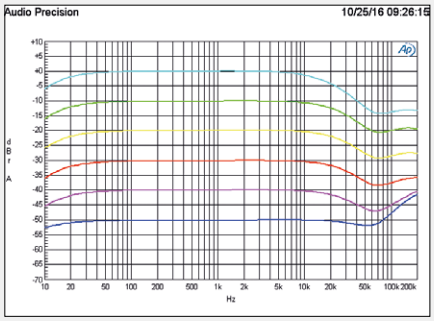


FFT bei 1 kHz und einer höheren Verstärkung (70 dB) und V_{Aus} = 1 V.
Da das Rauschen um einen Pegel von 90 dB auftritt, liegen die Werte „total harmonic distortion + noise“ bei 0,078 %.

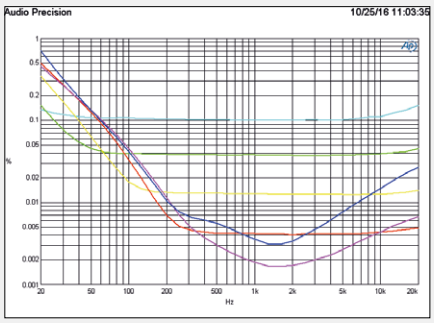
Tabelle 2. Messung mit **Pikatron ÜP3095M**

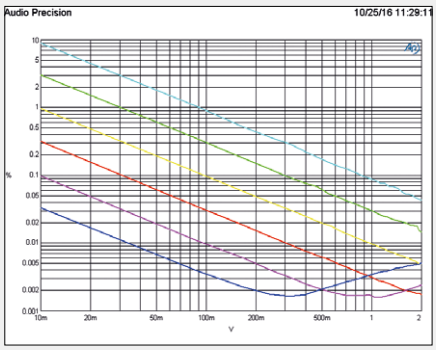
Gain	THD+N (Quellimpedanz 200 Ω) 1 kHz, 1 V_{Aus}, B 22 kHz	S/N (Quellimpedanz 200 Ω) in Bezug auf 1 V_{Aus}, B 22 kHz	Eckfrequenz bei -3 dB tief/hoch
20	0,0034%	109 dB / 112 dBA	< 10 Hz / ? >200 kHz)
30	0,0016%	100 dB / 102 dBA	14,5 Hz / 30 kHz
40	0,0031%	90,6 dB / 92,6 dBA	15,3 Hz / 28,3 kHz
50	0,0097%	80 dB / 82,2 dBA	15,3 Hz / 26,2 kHz
60	0,030%	70,4 dB / 72,5 dBA	15,3 Hz / 23,3 kHz
70	0,086%	61,1 dB / 62,7dBA	15,3 Hz / 16,2 kHz

Amplitude als Funktion der Frequenz für die verschiedenen Verstärkungseinstellungen, von 10 bis 200 kHz.
Es gibt eine Resonanzspitze direkt über 200 kHz. Die Gesamtbandbreite ist etwas niedriger als diejenige, die beim Lundahl-Trafo gemessen wurde, aber für die Praxis immer noch ausreichend. Wird C5 durch einen Wert von 22 µF ersetzt, so wird die Eckfrequenz von 15,3 auf 12 Hz gesenkt.

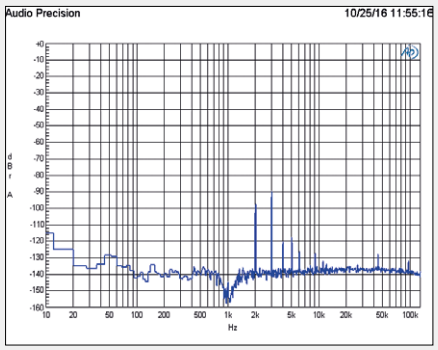


Distortion + noise als Funktion der Frequenz bei einer Ausgangsspannung von 1 V und verschiedenen Verstärkungs-Einstellungen bei einer Bandbreite von 80 kHz.
Nur bei den tiefsten Frequenzen sind die Werte von „total harmonic distortion + noise“ höher als diejenigen, die beim Lundahl-Trafo gemessen wurden.

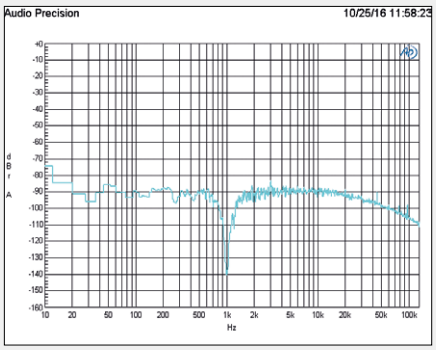




Distortion + noise als Funktion des Ausgangspegels bei 1 kHz und einer Bandbreite von 22 kHz.
Ein Unterschied zur entsprechenden Lundahl-Kurve ist hier nur schwer festzustellen.

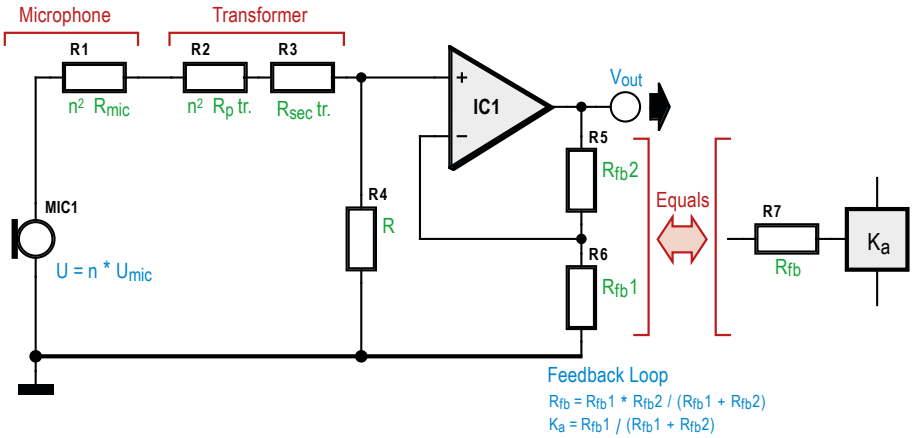


Fast Fourier-Transformation (FFT) bei 1 kHz für die geringste Einstellung der Verstärkung (20 dB) und V_{Aus} = 1 V.
Die dritte Harmonische tritt am deutlichsten hervor. Die restlichen Harmonischen sind etwas stärker als bei der mit Lundahl gemessenen Kurve. Der Wert „total harmonic distortion + noise“ beträgt hier 0,0034 %.



FFT bei 1 kHz und einer höheren Verstärkung (70 dB) und V_{Aus} = 1 V.
Die dritte Harmonische ist kaum sichtbar und es bleibt nur das Rauschen. Der Wert „total harmonic distortion + noise“ beträgt 0,086 %.

Analyse des Vorverstärker-Rauschens



Ersatzschaltbild zur Rauschanalyse des Vorverstärkers.

1. Theorie

Das betrachtete Rauschen ist zum Teil auf inhärentes, thermisches Rauschen der Widerstände zurückzuführen und wird gleichzeitig auch durch aktive Bauteile wie zum Beispiel den Operationsverstärker erzeugt. Weitere Störquellen wie elektromagnetische oder elektrostatische Felder wurden nicht in Betracht gezogen. Grundsätzlich lassen sich viele Störungen durch eine geeignete Abschirmung beheben.

Die **Abbildung** zeigt ein Ersatzschaltbild zur Rauschanalyse des Vorverstärkers. Das Rauschen stammt aus den beiden oben erwähnten Quellen. Nur die Realanteile der Impedanzen tragen zum Rauschen bei.

Die Signalquelle wird durch das Mikrofon repräsentiert, das einen internen Widerstand R_{mic} besitzt und das Signal U_{mic} über den Trafo TR1 an den positiven Eingang des Operationsverstärkers weitergibt. Jede Trafowicklung bildet einen internen Widerstand: Am Primär-Anschluss ist dies $\mathbf{R_{p\,tr}}$ und am Sekundär-Anschluss handelt es sich um $\mathbf{R_{sec\,tr}}$. Der Trafo wird durch das Übersetzungsverhältnis n zwischen Primär und Sekundär beschrieben.

Auf den positiven Opamp-Eingang bezogen ergibt sich der Gesamtwiderstand der aus Mikro und Trafo bestehenden Schaltungseinheit aus:

$$(1) \quad R_a = n^2 \cdot \left(R_{mic} + R_{p\,tr}\right) + R_{sec\,tr}$$

Dieser Widerstand befindet sich gegebenenfalls parallel zu R4, wenn dieser eingesetzt wird. In diesem Falle ergibt sich ein Gesamtwiderstand von:

(2) $$R_{inp\ pos} = \frac{1}{\frac{1}{R_a} + \frac{1}{R_4}} = \frac{1}{\frac{1}{n_2 \cdot \left(R_{mic} + R_{p\ tr}\right) + R_{sec\ tr}} + \frac{1}{R_4}}$$

Der negative Eingang des Opamps ist mit den Widerständen $R_{fb\ 1}$ und $R_{fb\ 2}$ verbunden. Dieser Gegenkopplungskreis ist durch eine Abschwächung Ka und einen Widerstand gekennzeichnet, der sich wie folgt ergibt:

(3) $$K_a = \frac{R_{fb_1}}{R_{fb_1} + R_{fb_2}}$$

und

(4) $$R_{fb} = \frac{1}{\frac{1}{R_{fb1}} + \frac{1}{R_{fb2}}}$$

An den Eingängen des Opamps ergibt sich das thermische Rauschen aus folgender Gleichung:

(5) $$e_{n\ th} = \sqrt{4 \cdot k_b \cdot T \cdot \left(R_{inp\ pos} + R_{fb}\right) \cdot B}$$

k_b = Boltzmann-Konstante
T = Temperatur in Grad Kelvin
B = Signalbandbreite

Die sich am Opamp ergebende Rauschspannung berechnet sich wie folgt:

(6) $$e_{n\ act\ comp} = \sqrt{\left\{e^2_{n\ op} + \left[i_{n\ op} \cdot \left(R_{inp\ pos} + R_{fb}\right)\right]^2\right\} \cdot B}$$

$e_{n\ op}$ und $i_{n\ op}$ entsprechen jeweils den in den Datenblättern der Bauelemente veröffentlichten Input-Voltage-Noise-Density- und Current-Noise-Density-Werten, aus denen sich der Gesamtwert der Rauschspannung wie folgt ergibt:

(7) $$e_{n\ gesamt} = \sqrt{e^2_{n\ th} + e^2_{n\ act\ comp}}$$

Da die Rauschquellen nicht korrelieren, kann die resultierende Rauschspannung nicht durch einfache Addition errechnet werden, sondern ergibt sich aus der Wurzel der Summe der Quadrate ihrer Amplituden, was bedeutet, dass hier individuelle Größen ins Spiel kommen.

2. Zahlenbeispiel

Wollen wir nun die Zahlen aus dem Schaltplan in Bild 1 zum Vergleich von vier Operationsverstärkern nutzen: OPA134 von Burr Brown, AD743 von Analog Devices, NE5534 und TL071. Betrachten wir das Ganze bei einer Vorverstärker-Bandbreite von 30 kHz und einer Umgebungstemperatur von 26,85 °C, die einem warmen, sonnigen Sommernachmittag entspricht. Das Trafo-Übersetzungsverhältnis soll den Wert 10 besitzen, bei einem Primärwiderstand von 50 Ω und einem Sekundärwiderstand von 1500 Ω. Nehmen wir an, das Mikrofon besäße einen internen Widerstand von 100 Ω, ein durchaus realistischer Wert. Der Rauschpegel wird beim größten Wert von R_{fb} berechnet, was einer maximalen Verstärkung entspricht.

Beim Prototyp des Vorverstärkers wird Ra nicht verwendet. Das thermische Rauschen der Widerstände wird mittels der Gleichungen (1), (4) und (5) berechnet:

$$R_{inp\ pos} = 10^2 \cdot (100 + 50) + 1500 = 16500$$

und

$$R_{fb} = \frac{1}{\frac{1}{1000} + \frac{1}{319 \cdot 10^3}} = 996{,}87 \approx 997$$

Die Werte werden in Gleichung (5) eingesetzt:

$$e_{n\ th} = \sqrt{4 \cdot 1{,}38 \cdot 10^{-23} \cdot 300 \cdot (16500 + 997) \cdot 30000}$$

Daraus ergibt sich:

$$e_{n\ th} = 2{,}95 \cdot 10^{-6}\ \text{V}$$

Der Rauschanteil des Verstärkers wird mit Gleichung (6) berechnet, und sein gesamter Wert mittels Gleichung (7) auf den Eingang zurückgeführt. Die Tabelle zeigt die Unterschiede zwischen den einzelnen Opamps in recht übersichtlicher Weise. Die letzte Zeile der Tabelle gilt für eine Mikrofonspannung von 5 mV.

Opamp	AD743	OPA134	NE5534	TL 071
$e_{n\ op}$ (nV / √Hz)	2,9	8,0	4,0	18,0
$i_{n\ op}$ (pA / √Hz)	0,0069	0,003	1,5	0,010
$e_{n\ act\ comp}$	502 nV	1,39 µV	5,24 µV	3,12 µV
$e_{n\ gesamt}$	2,99 µV	3,26 µV	5,47 µV	4,29 µV
(S+N)/N (dB)	84,46	83,72	79,23	81,33

Mit Ausnahme des NE5534 sind alle Opamps mit FET-Eingängen versehen und zeichnen sich durch ein vernachlässigbares Eingangsrauschen aus. Die mit dem OPA134 (Bild 1) erzielten Ergebnisse stellten sich als besonders positiv heraus. Auch die Leistungen des TL071 sind sehr erstaunlich, wogegen der NE5534 in der vorliegenden Schaltung durch sein Stromrauschverhalten eher enttäuscht, was jedoch nicht bedeutet, dass er in anderen Anwendungen nicht gute Leistungen vollbringt.

3. Einfluss des Trafos auf das thermische Rauschen

Gleichung (1) kann auch wie folgt geschrieben werden:

(8) $$R_a = n^2 \cdot R_{mic} + n^2 \cdot R_{p\ tr} + R_{sec\ tr}$$

Es ist ersichtlich, dass diese Gleichung zwei Terme enthält: Einen für das Mikrofon und einen für den Trafo, für den man einen äquivalenten, auf die Sekundärwicklung bezogenen Widerstand definieren kann:

(9) $$R_{eq\ tranfo} = n^2 \cdot R_{p\ tr} + R_{sec\ tr}$$

Der Beitrag von Mikro und Trafo zum thermischen Rauschen ergibt sich demnach wie folgt:

(10) $$e_{n\ th\ mic\ und\ trafo} = \sqrt{4 \cdot k_b \cdot T \cdot \left(n^2 \cdot R_{mic} + \mathrm{R}_{eq\ trafo}\right) \cdot B}$$

Der Trafo multipliziert die Mikrofon-Eingangsspannung U_{mic} mit dem Faktor n. Die Berechnung des Signal-Rausch-Verhältnisses an der Sekundärwicklung ergibt demnach:

$$(11) \quad \frac{S+N}{N} = \frac{n \cdot U_{mic} + \sqrt{4 \cdot k_b \cdot T \cdot \left(n^2 \cdot R_{mic} + R_{eq\,trafo}\right) \cdot B}}{\sqrt{4 \cdot k_b \cdot T \cdot \left(n^2 \cdot R_{mic} + R_{eq\,trafo}\right) \cdot B}}$$

Zum Vergleich betrachten wir dieselbe Berechnung nun für einen Trafo ohne inneren Widerstand:

$$(12) \quad R_{eq\,trafo} = 0$$

Nach Vereinfachung ergibt sich folgende Gleichung (11):

$$(13) \quad \frac{S+N}{N} = \frac{U_{mic} + \sqrt{4 \cdot k_b \cdot T \cdot R_{mic} \cdot B}}{\sqrt{4 \cdot k_b \cdot T \cdot R_{mic} \cdot B}}$$

Vergleichen wir die Gleichungen (11) und (13): Ein Trafo ohne inneren Widerstand wird das Signal-Rauschverhältnis der Mikrofonspannung weder verbessern noch verschlechtern. Im Falle eines realen Trafos können diese Widerstände zur Erhöhung des Rauschpegels beitragen, so dass die Wahl der Bauteile hier einer besonderen Aufmerksamkeit bedarf

Weblinks

[1] www.analog.com

[2] www.ti.com

[3] www.buerklin.com

[4] www.lundahl.se/microphone/

[5] www.sowter.co.uk/pro-audio-transformers.php#

[6] www.jensen-transformers.com/mic_in.html

[7] Mikrofonaufsätze von Jörg Wuttke: www.schoeps.de/documents/Mikrofonbuch_komplett.pdf

[8] Norm EN 61938: https://de.wikipedia.org/wiki/Phantomspeisung

[9] Projektseiten: www.elektormagazine.de/140426 www.elektormagazine.de/labs/1161

Anzeige

Review:
USB-Mikroskop Andonstar V160

Für die Arbeit mit SMDs mehr als nützlich

Der manuelle Aufbau von Prototypen mit SMDs kann schnell zu einem Geduldsspiel werden, denn viele SMDs haben so winzige Dimensionen, dass Lötfehler nicht ohne Mikroskop behebbar sind. Außer den konventionellen Mikroskopen stehen so genannte USB-Mikroskope zur Wahl, sie zeigen das Bild auf dem Schirm eines PCs oder Tablets an. Im Elektor-Labor haben wir das USB-Mikroskop Andonstar V160 getestet.

Von **Luc Lemmens** (Elektor-Labor)

Eine wacklige Lötstelle, eine Lötbrücke zwischen zwei Pins: Ohne maschinelles Lötgerät lassen sie sich kaum vermeiden. Eine starke Lupe oder Lupenleuchte kann helfen, solche Missgeschicke aufzuspüren. Doch um dem Fehler auf die Spur zu kommen, reicht oft die Vergrößerung nicht aus. Der Abstand zwischen Lupe und Lötstelle ist häufig so schmal, dass die Lötkolbenspitze nicht dazwischen passt. Für solche misslichen Lagen haben die Hersteller optischer Geräte professionelle Hilfen im Angebot. Die Palette reicht von hochklassigen Stereo-Mikroskopen bis zu auflösungsstarken Inspektionskameras oder sogar speziellen Röntgensystemen. Professionell sind natürlich auch die Preise, die in den Angebotskatalogen stehen. Eine kostengünstige Alternative stellen die USB-Mikroskope dar. Die Modellvielfalt ist so riesig, dass die Wahl nicht leicht fällt.

Stabiles Stativ: Unverzichtbar!

Ausschlaggebend für die Entscheidung sind bei einem USB-Mikroskop nicht allein die Megapixel oder die Vergrößerung. Auch

die mechanischen Eigenschaften haben maßgeblichen Einfluss auf die Tauglichkeit für den beabsichtigten Zweck. Kostengünstige USB-Mikroskope werden meistens mit wenig stabiler Mechanik ausgestattet, so dass schon geringe Erschütterungen des Arbeitstisches oder ein leichtes Berühren der Kamera genügen, um das Bild oder die Fokussierung ins Wanken zu bringen. Das USB-Mikroskop Andonstar V160 ist eine löbliche Ausnahme. Das massive Stativ gewährleistet ein stabiles und scharfes Bild, die Parameter lassen sich präzise einstellen. Der Bildausschnitt hängt von Stellschrauben am Stativ ab, und für das Scharfeinstellen ist ein drehbarer Ring am Kameragehäuse zuständig. Die Bauweise der Kamera ist klein und schmal, sie liefert auch dann scharfe Bilder, wenn der Abstand zum Objekt etliche Zentimeter beträgt. Dann bleibt genügend Raum, um mit dem Bildschirm im Blick den Lötkolben zu bewegen. Bezogen auf das Objekt kann das Andonstar V160 auch winklige Positionen einnehmen, so dass die Bauteile seitlich betrachtbar sind. In solchen Positionen ist noch mehr Bewegungsfreiheit für die Arbeitsgeräte vorhanden.

Für PC und Tablet

USB-Mikroskope sind mit elektronischen Kameras eng verwandt, sie sind mit den meisten von aktuellen Betriebssystemen unterstützten Kameraprogrammen kompatibel. Zum Lieferumfang des Andonstar V160 gehört eine CD, auf der sich allerdings nur Windows-Software befindet. Die Website von Oasis Scientific [1] hält auch Software für andere gängige Betriebssysteme bereit – ausgenommen ist jedoch Linux!

Wir haben dieses USB-Mikroskop an einem Android-Tablet erprobt, eine sinnvolle Kombination, falls der Platz auf dem Arbeitstisch begrenzt ist. Die App, die wir genutzt haben, heißt mScope, sie lief zu unserer vollen Zufriedenheit. Für mScope muss das Tablet USB-OTG unterstützen, und für den Anschluss ist ein USB-OTG-Adapterkabel erforderlich. Die meisten Kamera-Apps können Fotos oder sogar Filmclips speichern, beim Dokumentieren von Projekten kann das oft nützlich sein.

Integrierter LED-Ring

Das Andonstar V160 verfügt über eine ringförmig um das Objektiv angebrachte LED-Beleuchtung, die Helligkeit lässt sich mit einem Steller im Anschlusskabel steuern. Damit ist zwar das schattenfreie Ausleuchten kleiner Objekte möglich, doch es können auch störende Spiegelungen auftreten, falls die Oberfläche das Licht reflektiert. Als Ursache unerwünschter Lichtreflexionen rangieren glänzende Lötinseln oder auch Teile der Platinenoberfläche an erster Stelle. In solchen Fällen empfiehlt es sich, den LED-Ring abzuschalten und eine diffuse Lichtquelle neben dem Mikroskop zu platzieren.

Inzwischen unverzichtbar

Nach Anschaffen und Testen des Andonstar V160 werden im Elektor-Labor nur noch selten Lupen oder Lupenleuchten eingesetzt, wenn es um das Löten von SMDs geht. Das Koordinieren der Werkzeugbewegungen mit der Darstellung auf dem Bildschirm bedarf allerdings etwas Übung. Bei maximaler Auflösung (1600 · 1200 Pixel) tritt eine leichte Verzögerung ein, auch daran muss sich der Benutzer erst gewöhnen. Ferner kann es nicht schaden, den aufgeheizten Lötkolben auch außerhalb des Bildschirms im Auge zu behalten, um Sengschäden an der Kamera zu vermeiden. Insgesamt ist der Umstieg auf die indirekte Objektbetrachtung über den Bildschirm nicht schwierig. An unserem SMD-Arbeitsplatz hat sich das USB-Mikroskop Andonstar V160 inzwischen in den etablierten Werkzeugbestand eingereiht. ⏮

(160367)gd

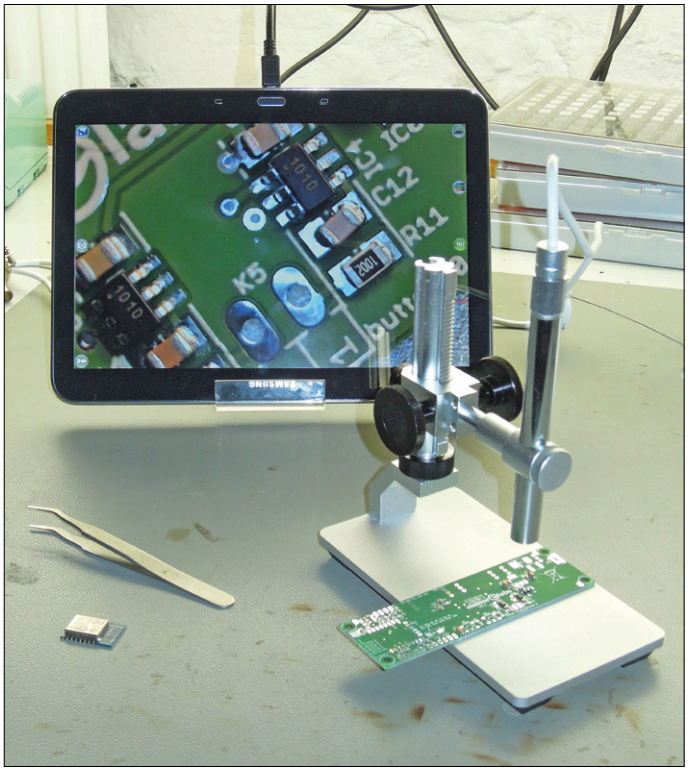


Bild 1. Das Andonstar V160 zusammen mit einem Android-Tablet in Aktion.

> Zitterfreie Bilder nur mit stabilem Stativ!

Weblinks

[1] www.oasisscientific.com/downloads.html

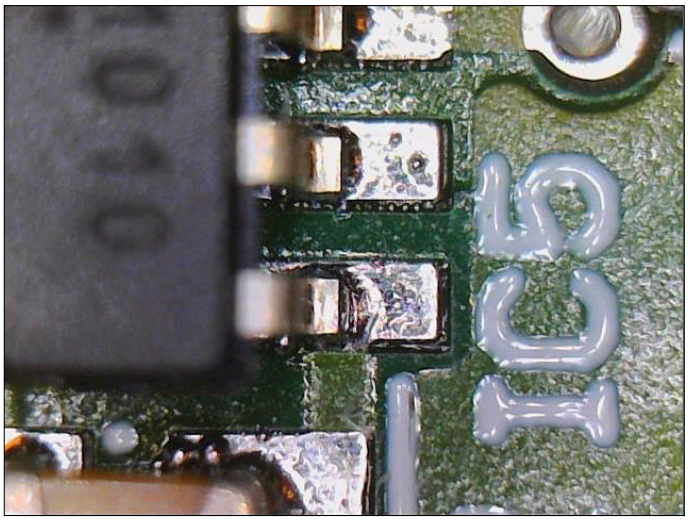


Bild 2. Die SMD-IC-Anschlüsse stark vergrößert auf dem Bildschirm.

Der I²C-Bus

Teil 1: Protokoll

Von **Josef Möllers**

Der Inter-IC- oder kurz I²C-Bus ist ein 2-Draht-Bus, an den sich einfach Peripherie anbinden lässt, wenn nur wenige Daten (und dies nicht allzu schnell) übertragen werden sollen. In einer dreiteiligen Serie erklären wir die Grundzüge und zeigen, wie man den Bus beim Raspberry Pi, Arduino und anderen Systemen praktisch einsetzt.

Diese kleine, dreiteilige Artikelreihe beschreibt den I²C-Bus (manchmal, zum Beispiel von Atmel, auch als Two-Wire-Interface TWI bezeichnet) nicht bis ins letzte Detail, sondern soll zeigen, wie einfach der Bus in einen Aufbau zu integrieren und wie unkompliziert er anzusteuern ist. Zunächst erfahren wir etwas über das Datenprotokoll, dann, im zweiten Teil, wollen wir den I²C-Bus an verschiedenen Systemen – vom Arduino und Raspberry Pi bis hin zum ausgewachsenen PC - ausprobieren und schließlich zum Abschluss einige typische I²C-Slaves kennenlernen und Wege zeigen, wie man Fehlern in der Datenübertragung auf die Spur kommen kann.

Der I²C-Bus wurde in den 1980er Jahren von Philips entwickelt. Wer den I²C-Bus nicht nur als Blackbox mit fertigen Software-Bibliotheken verwenden, sondern wirklich bis auf Bitebene verstehen will, dem sei ergänzend die offizielle Spezifikation von NXP zum Download und natürlich auch zur Lektüre empfohlen [1].
Es gibt im Internet eine Unzahl von Ressourcen zum Thema I²C-Bus. Als Beispiel sei hier die englischsprachige Webseite [2] des Hamburger Systemhauses Telos genannt, die nahezu alle Aspekte der Kommunikation über diesen Bus behandelt.

Ein wenig Physik

Im Regelfall steuert man mit einem programmierbaren Mikrocontroller einen oder mehrere Peripheriebausteine mit fester Funktion, wie Porterweiterungen, Uhren-Chips oder EEPROMs an. Der Controller ist der Master, der den Takt vorgibt (siehe unten), die Peripherie-ICs sind die Slaves. Via I²C können sich aber auch zwei Mikrocontroller miteinander unterhalten, wobei

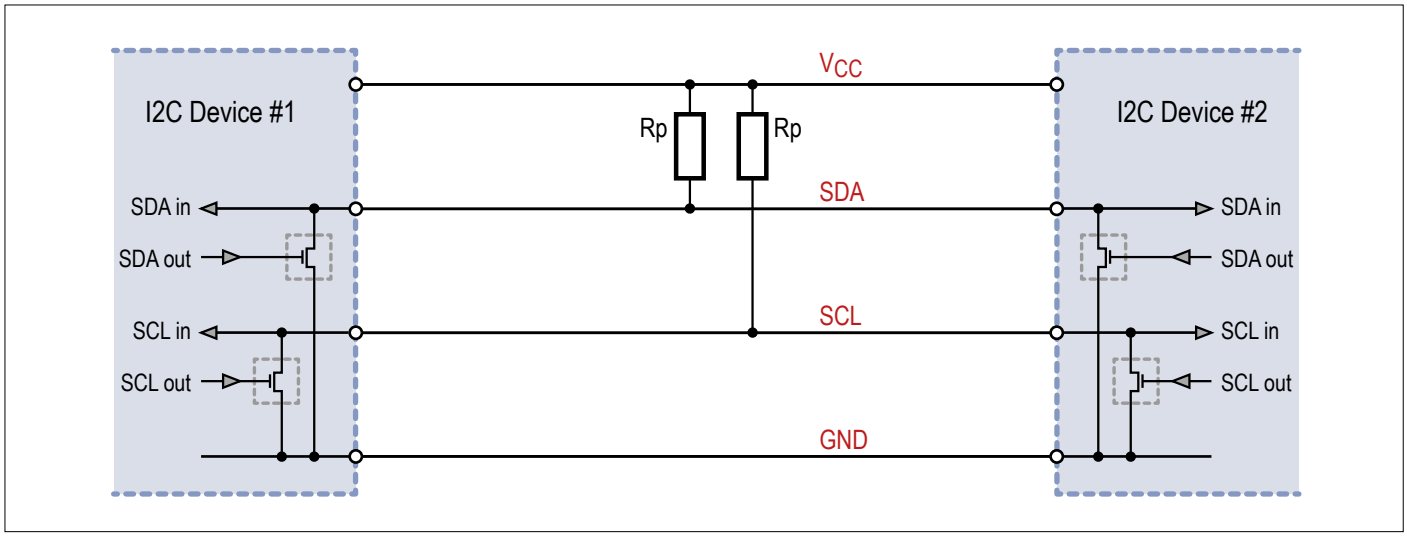


Bild 1. Der I²C-Bus stellt eine Wired-AND-Verbindung dar.

dann einer die Master-Funktion übernimmt. Werden mehrere Slaves angesteuert, dann hängen alle Bausteine hintereinander an denselben Leitungen (Bus). An einem I²C-Bus können auch mehrere Master angeschlossen sein.

Die beiden Drähte des I²C-Busses transportieren ein Taktsignal (SCL) und ein Datensignal (SDA). So lassen sich bereits mit einem schmalen 4-adrigen Flachbandkabel Peripheriebausteine ansteuern und gleichzeitig mit Energie versorgen. Um Übersprechen zu minimieren, legt man die Masseleitung am besten zwischen die beiden Signalleitungen.
Beide Leitungen haben positive Logik: Hohes Potential auf SDA signalisiert eine „1“ und für einen Taktimpuls wird SCL kurzzeitig auf ein hohes Potential gelegt.

Im Ruhezustand ziehen Pull-Up-Widerstände die beiden Leitungen auf ein hohes Potential. Die Treiber in I²C-Bausteinen besitzen einen offenen Kollektor oder offenen Drain, so dass beide Leitungen jeweils ein großes Wired-AND darstellen (**Bild 1**). Die Leitung beziehungsweise der gemeinsame Punkt ist nur dann HIGH, wenn **alle** angeschlossenen Ausgänge HIGH sind. Ist auch nur ein Ausgang LOW, so folgt auch die gemeinsame Leitung auf LOW. Jeder Busteilnehmer muss die Signalleitungen daher auch kontrollieren: Will er eine „1“ anlegen, muss er ausschließen können, dass ein anderer Teilnehmer gleichzeitig eine „0“ angelegt hat.

Die Größe der Widerstände ist relativ unkritisch und sollte einige wenige Kiloohm betragen. Beim Raspberry Pi sind Pull-Up-Widerstände (1,8 kΩ nach 3,3 V) bereits fest eingebaut und immer aktiv. Bei den Atmel-Mikrocontrollern der ATmega-Serie sind Pull-Up-Widerstände per Software zuschaltbar, man sollte jedoch davon keinen Gebrauch machen, da sie mit 20...50 kΩ zu groß dimensioniert sind.

Laut Spezifikation sollten der HIGH-Pegel mindestens 0,7 V_{DD} und der LOW-Pegel höchstens 0,3 V_{DD} betragen, in der Praxis funktionieren 5-V-Bausteine auch mit Pull-Up-Widerständen nach 3,3 V (beispielsweise an einem Raspberry Pi). Sollte eine Schaltung aber einmal nicht funktionieren, können verschiedene Pegel der Grund sein und man muss dann Pegelumsetzer einbauen.

Es gibt auch I²C-Slave-Module mit eingebauten Pull-Up-Widerständen. Diese sollte man aber entfernen oder bei einem Bausatz erst gar nicht bestücken, denn wenn mehrere Slaves ihre eigenen Pull-Up-Widerstände mitbringen, liegen diese parallel und der resultierende Gesamtwiderstand wird zu klein. An einem I²C-Bus sollte es nur **ein** Paar Pull-Up-Widerstände geben, am besten nur auf oder an dem Master!
Hier ergibt sich aber gleich ein Problem: Einige Bausteine, die mit 3,3 V betrieben werden, vertragen an ihren Eingängen keine höheren Spannungen. Wenn man die Pull-Up-Widerstände nun mit 5 V verbindet, werden diese Bauteile eventuell zerstört! Andererseits muss man, damit der Bus zuverlässig arbeitet, die vorgeschriebenen Pegel einhalten. Damit 5-V- und 3,3-V-Komponenten an einem gemeinsamen Bus arbeiten können, sollte man in diesem Fall bidirektionale Pegelumsetzer (Level-Shifter) wie in [3] beschrieben einsetzen. Dabei verbinden zwei MOSFETs und zusätzliche Pull-Up-Widerstände den 5-V- mit dem 3,3-V-Abschnitt eines I²C-Busses (**Bild 2**).

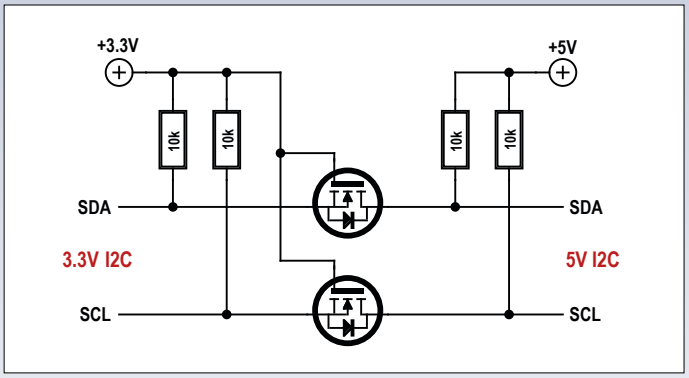


Bild 2. MOSFETs teilen den Bus in einen 5-V- und einen 3,3-V-Abschnitt auf.

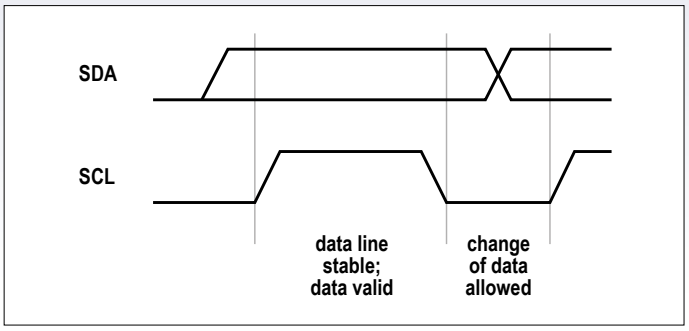


Bild 3. Timing einer Datenübertragung.

Das Protokoll

Der I²C-Bus ist ein reiner Master-Slave-Bus. Das bedeutet, dass die Kommunikation **immer** von einem Master ausgeht, der einen *Slave* adressiert und dann mit ihm Daten austauscht. Ein Slave kann nicht von sich heraus eine Kommunikation einleiten, wie es zum Beispiel bei SCSI möglich ist (*Asynchronous Event Notification* oder *Reselect*).
Wie schon oben erwähnt, können zum Beispiel Mikrocontroller sowohl eine Master- als auch eine Slave-Funktion ausüben. Die Funktion kann theoretisch während des Betriebs auch wechseln. Meist ist ein Teilnehmer aber doch entweder nur Master oder nur Slave.

Der Master bestimmt immer die Geschwindigkeit der Übertragung. Es werden offiziell vier Geschwindigkeiten unterstützt:

- 100 kbps (Standard Mode, Sm, default),
- 400 kbps (Fast Mode, Fm),
- 1 Mbps (Fast Mode Plus, Fm+) und
- 3,4 Mbps (High Speed Mode, Hs-mode).

Sowohl der Master als auch der Slave können Daten an den jeweils anderen übertragen, lediglich der Takt wird dabei immer vom Master auf SCL vorgegeben. Die Datenübertragung (**Bild 3**) erfolgt dabei taktsynchron: Um ein Bit zu übertragen, legt der Sender das Datenbit in positiver Logik auf SDA. Der Master gibt die Taktleitung SCL frei, die daraufhin auf HIGH gezogen wird, und zieht sie nach einiger Zeit zurück auf LOW. Damit ist ein Bit übertragen und der Sender kann das nächste Bit anlegen. Wenn SCL während der Übertragung HIGH ist, darf SDA sich nicht verändern.

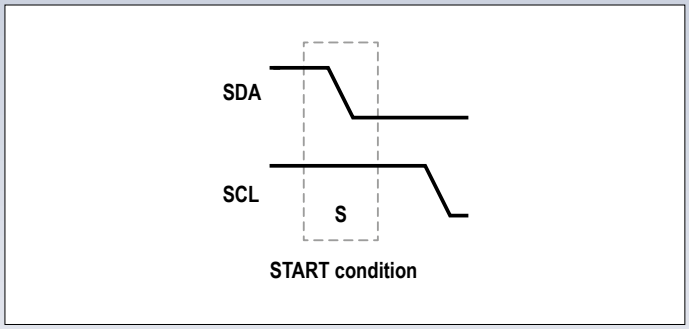


Bild 4. Startbedingung.

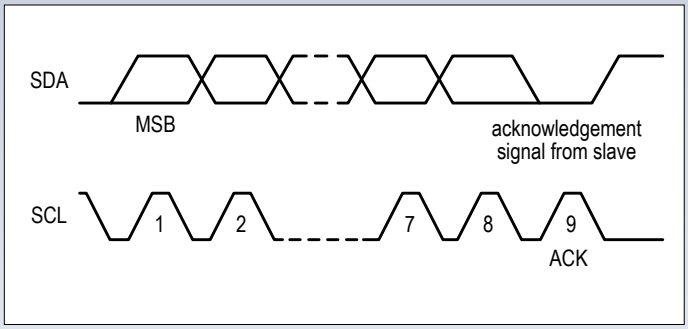


Bild 5. Übertragung des Adressbits.

1. Um die Übertragung einzuleiten, zieht der Master erst die Datenleitung SDA und dann die Taktleitung SCL auf LOW (**Bild 4**). Diese Sequenz heißt *START condition*.

2. Nach der *START condition* sendet der Master wie in **Bild 5** zu sehen zuerst die Slave-Adresse (höchstwertiges Bit MSB zuerst), gefolgt vom Schreib-/Lese-Bit (0 = Daten vom Slave zum Master oder 1 = Daten vom Master zum Slave). Die Slave-Adresse ist in der Regel sieben Bit lang (der I²C-Standard kennt auch längere 10-bit-Adressen, was aber in der Praxis nur selten genutzt wird) und wird vom Hersteller des Peripheriebausteins ganz oder teilweise vorgegeben. Meist kann man über externe Beschaltung einige Adressbits selber festlegen, so dass sich mehrere gleiche Bausteine am Bus mit unterschiedlichen Adressen ansprechen lassen. Die Adresse eines Peripheriebauteils ist im Datenblatt angegeben oder lässt sich mit einem Tool wie `i2cdetect` ermitteln. Zusammen sind Slave-Adresse und Schreib-/Lese-Bit acht Bit, also ein Byte lang.

3. Nach der Adresse werden im SCL-Takt des Masters die Nutzdaten übertragen, immer byteweise mit dem höchstwertigen Bit (MSB) zuerst (**Bild 6**). Wenn das Schreib-/Lese-Bit im Adressbyte „1“ war, erwartet der Master, dass der Slave die Datenbits rechtzeitig auf SDA legt, bevor er selber den Takt-Impuls auf SCL erzeugt. War das Schreib-/Lesebit dagegen „0“, legt der Master selber erst die Datenbits auf SDA, um dann auch den Takt-Impuls zu erzeugen. Jedes Byte (auch das Byte, das die Adresse und das Schreib-/Lese-Bit enthält) muss vom jeweiligen Empfänger mit einem neunten Bit quittiert werden. LOW steht für eine positive Quittung (ACK), HIGH für eine negative (NACK). Ein NACK bedeutet übrigens nicht, dass die Übertragung oder die Daten fehlerhaft waren, sondern nur, dass das quittierte Byte das letzte der Übertragung war. Beim Adressbyte zeigt das Fehlen des ACK an, dass kein Slave die Adresse erkannt hat oder dass die Kombination aus Adresse und Schreib-/Lesebit ungültig ist.

4. Am Ende der Übertragung lässt der Master erst die Taktleitung SCL und dann die Datenleitung SDA los, die dann beide daraufhin von den Pull-Ups wieder auf HIGH gezogen werden. Diese Sequenz (**Bild 7**) heißt *STOP condition*. Gegebenenfalls muss der Master dazu erst die Datenleitung SDA auf LOW ziehen.

START condition und *STOP condition* sind Ausnahmen von der Regel, dass SDA sich nicht verändern darf, wenn SCL auf HIGH liegt.

Die Anzahl der Bytes, die in einer Transaktion übertragen werden kann, ist eigentlich unbegrenzt. Höhere Protokolle wie der *Power Management Bus* (PMBus) oder der *System Management Bus* (SMBus), die auf den I²C-Bus aufsetzen, legen aber Obergrenzen dafür fest oder definieren Record-Formate, die zum Beispiel eine Längenangabe für die Übertragung enthalten. Allerdings ist die „grenzenlose“ Übertragung nicht mit jeder Hardware oder mit jeder Bibliothek möglich, weil manchmal die Anzahl der zu empfangenden Bytes schon beim Start einer Übertragung festgelegt sein muss. Ein Beispiel dafür ist die Wire-Bibliothek des Arduino. Auch definieren diese Protokolle Timeout-Werte (beispielsweise für das *Clock-Stretching,* siehe unten) und eine Fehlererkennung über Redundanzprüfung (*Packet Error Checking*, PEC).
Eine Transaktion (zwischen *START condition* und *STOP condition*) ist immer unidirektional. Mit dem Schreib-/Lese-Bit legt der Master die Übertragungsrichtung für die gesamte Transaktion fest.

Nur wenige Bausteine sind intern so einfach strukturiert, dass sie mit einer einfachen Übertragung auskommen (zum Beispiel der PCF8574, der im letzten Teil des Artikels behandelt wird). Bei den meisten I²C-Bauteilen muss der Master vor dem Lesen in einer Schreib-Transaktion einen Parameter, etwa eine Registernummer oder eine Speicheradresse, an den Slave senden. Will er danach Daten an den Slave schicken, kann er diese nach dem Parameter in derselben Schreib-Transaktion mit übertragen. Will der Master aber Daten vom Slave lesen, muss er eine neue, eine Lese-Transaktion beginnen. Um zu verhindern, dass sich ein zweiter Master in diese Abfolge drängt und demselben Slave einen anderen Parameter übermittelt, kann der aktive Master statt der Sequenz „STOP – START“ unmittelbar wieder eine *START condition* senden, diese nennt man dann *REPEATED START condition*:

```
Start – Adresse+Schreiben – Registernummer –
(Repeated) Start – Adresse+Lesen – Daten – … – Stop
```

Manche Peripheriebausteine wie Speicher oder die RTC RV-8523 erhöhen die Registernummer beziehungsweise die Speicheradresse nach jedem Zugriff automatisch, andere wie der

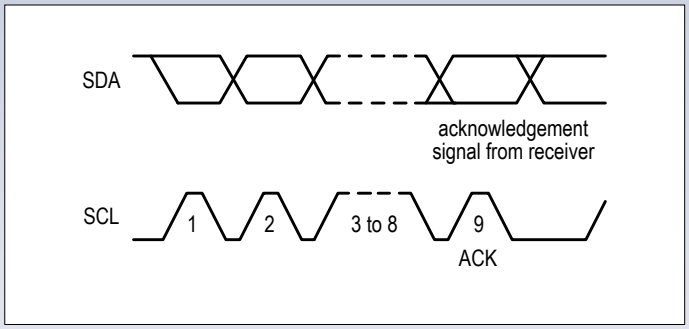


Bild 6. Übertragung eines Datenbits.

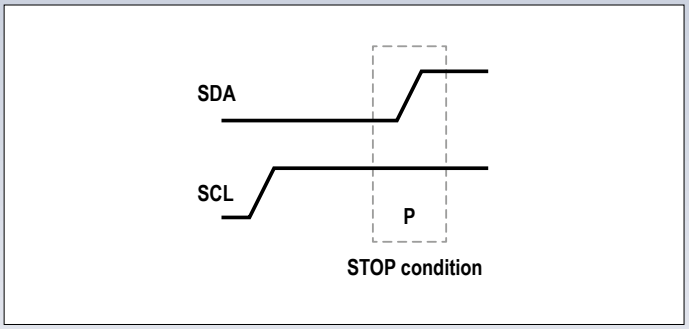


Bild 7. Stoppbedingung.

LM75-Temperatursensor tun dies nicht. Im ersten Fall kann man in einer Transaktion ganze Registersätze oder Speicherbereiche übertragen, im zweiten Fall muss man immer wieder neu adressieren. Andererseits: Will man ständig dasselbe Register lesen (zum Beispiel das Temperatur-Register des LM75), kann man sich die wiederholte Übertragung der Registernummer sparen. Details dazu entnehme man dem Datenblatt.

Interrupts sind im I²C-Protokoll nicht vorgesehen. Natürlich kann ein Slave über eine separate Leitung bei einem Master einen Interrupt auslösen, woraufhin dieser dann den Slave über die I²C Schnittstelle anspricht, der I²C Bus und das Protokoll selber definieren dafür aber keinen Standard.

Clock Stretching und Arbitration

Wenn ein Slave Daten übertragen soll, kann es einige Zeit dauern, bis der Baustein dazu bereit ist. Der Slave muss in dieser Wartezeit den Takt des Masters irgendwie stoppen oder verzögern. Da die beiden Bus-Leitungen wie ein großes Wired-AND funktionieren, kann der Slave dies veranlassen, indem er SCL für einige Zeit auf LOW festhält. Der Master erkennt dies, weil SCL nicht auf HIGH geht, wenn er die Leitung loslässt und wartet. Dieser Mechanismus wird *Clock Streching* genannt (streng genommen ist es eher ein *Pause Stretching*, da der Slave nur die Pause zwischen zwei Takt-Impulsen verlängern kann, nicht den Takt-Impuls selber).
An einem I²C-Bus können wie oben schon erwähnt auch mehrere Master angeschlossen sein. Zwischen Start- und Stopp-Bedingung gilt der I²C-Bus als belegt, kein anderer Master darf in dieser Zeit eine *START condition* anlegen. Ein (potentieller) Master muss den Bus also ständig im Auge behalten. Ein Master darf ja nur dann eine *START condition* senden, wenn der Bus frei ist. Sollten zwei Master aber zufällig gleichzeitig die *START condition* anlegen, sind meist entweder die beiden Slave-Adressen oder spätestens die Schreib-/Lese-Bits unterschiedlich. Dann wird der eine Master irgendwann eine „1“ senden wollen und der andere eine „0“. Wegen der Wired-AND-Funktionalität würde der Master, der eine „0“ senden will, diese auch auf SDA wiederfinden, während der andere Master seine „1“ nicht sieht und daraufhin sofort die Übertragung einstellen muss. Der „gewinnende“ Master (der die „0“ sendet) merkt davon gar nichts und fährt mit der Übertragung fort.

Man liest in diesem Zusammenhang (auch in der offiziellen I²C-Schrift) gelegentlich von „arbitration“ oder in deutschen Schriften „Arbitrierung“. Dies ist eigentlich falsch, denn es gibt keinerlei entsprechende separate Phase oder etwas Ähnliches mit Prioritäten und Timeouts. Wer etwas zu übertragen hat, wartet darauf, dass der Bus frei ist, fängt dann einfach an und horcht gleichzeitig, wie sich der Bus verhält. Nichts anderes wird im Kapitel „Arbitration“ in der NXP-Schrift beschrieben.

Ein Master kann übrigens durch eine *STOP condition* jederzeit eine Transaktion vorzeitig abbrechen. Auch ein Empfänger kann dies durch die Quittierung eines Bytes mit NACK erreichen, niemals aber ein Slave als Sender. Wenn nämlich ein empfangender Master zum Beispiel aufgrund eines Programmierfehlers weder ein Byte mit NACK quittiert noch die *STOP condition* sendet, wird die Transaktion nicht beendet. Der Slave kann seinerseits zwar aufgehört haben, Daten bereit zu stellen, der Master wird aber unbeirrt allen Unsinn, der sich dann eventuell auf SDA befindet, einlesen.

Im zweiten Teil

Im folgenden Teil des Artikels werden verschiedene Mikrocontroller, vom ATmega und Arduino über den Raspberry Pi bis hin zum PC, als I²C-Master und -Slave eingesetzt. Softwarebeispiele in C und Python zeigen, wie man Busteilnehmer adressiert, ihnen Befehle erteilt und sie ausliest. ◀

(160095)

Weblinks

[1] www.nxp.com/documents/user_manual/UM10204.pdf
[2] www.i2c-bus.org/
[3] http://playground.arduino.cc/Main/I2CBi-directionalLevelShifter

NCSA II – verbesserter Network Connected Signal Analyzer

Schneller, neue Fähigkeiten und einfacherer Einsatz

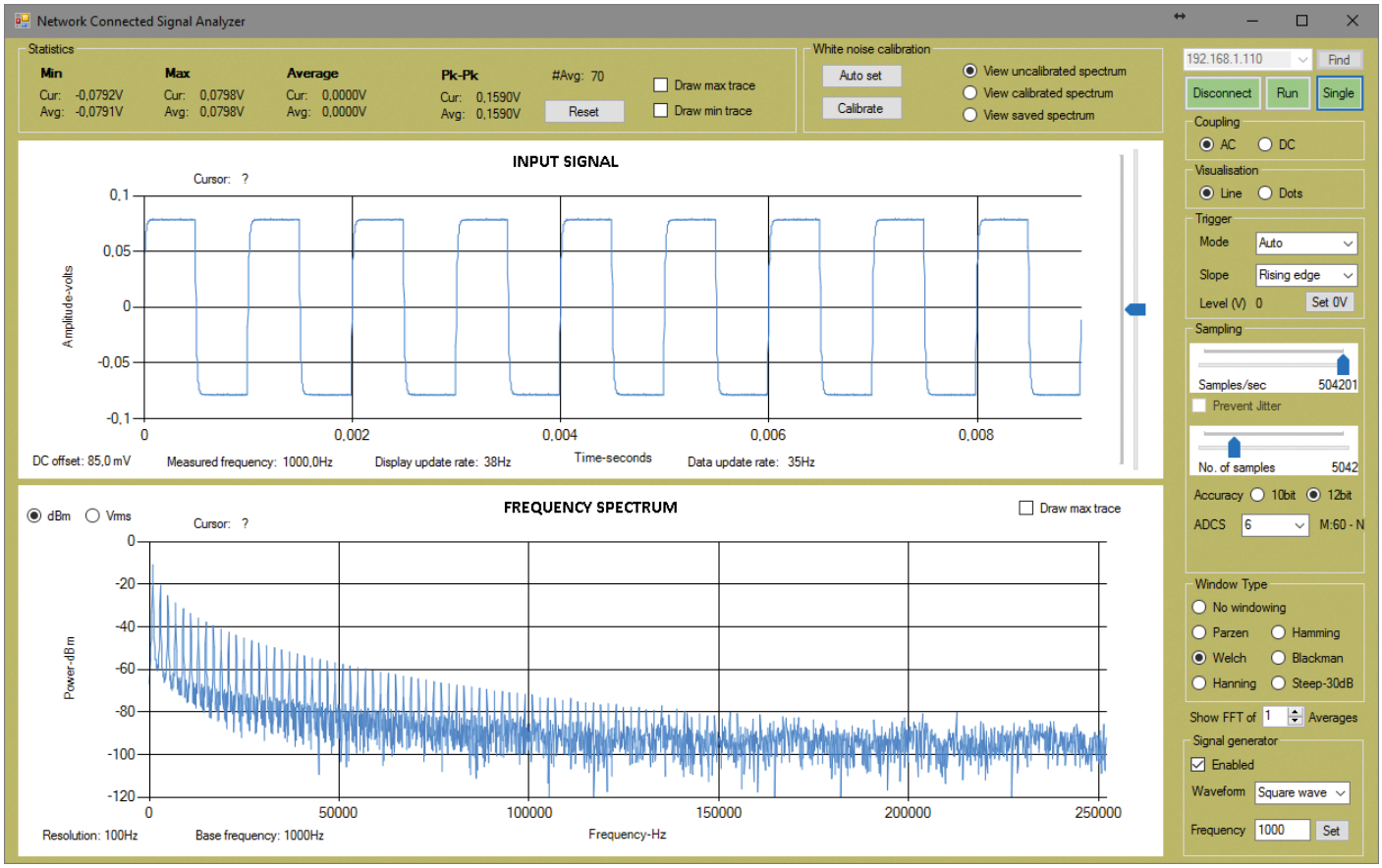


Bild 1. Das neue UI des verbesserten Signal-Analyzers.

In diesem Artikel geht es um ein Software- und Firmware-Upgrade des NCSA aus Elektor März und April 2016. Die hier vorgestellten Upgrades sind weit mehr als ein bloßes Update, denn sie ermöglichen deutliche Verbesserungen in Sachen Anschluss, Tempo, Funktionen und Einsatz – wobei die Hardware unverändert bleibt. Willkommen beim NCSA II!

Von **Joost Breed** (NL) und
Neal Martini (USA)

Zunächst ein kurzer Abriss zu den vier Aspekten, in denen sich Verbesserungen ergeben:

Verbesserte Netzwerkfähigkeit
NCSA II beherrscht jetzt PnP (**P**lug a**n**d **P**lay) beim Anschluss an ein Netzwerk (LAN). Man kann nun sogar zwei NCSA II gleichzeitig im gleichen lokalen Netz betreiben. Beides wurde möglich dank DHCP (Dynamic Host Configuration Protocol); beim Anschluss ans Netzwerk wird dem Gerät automatisch eine IP-Adresse vom DHCP-Server zugewiesen.

Mehr Tempo
NCSA II läuft jetzt nicht nur ein bisschen, sondern um ein Mehrfaches schnel-

ler als die erste Version. Es ergibt sich ein deutlich schnelleres Oszilloskop mit höheren Update-Raten der FFT (**F**ast **F**ourier **T**ransformation). Für die Verbesserungen beim PC-Client ist vor allem Multithreading verantwortlich, beim dsPIC33-Server kam SPI-DMA (**D**irect **M**emory **A**ccess) ins Spiel.

Mehr Funktionen
Das Oszilloskop wurde erweitert und der SWA (**S**ynthetic **W**aveform **A**nalyzer) bekam UDFs (**U**ser **D**efined **F**unctions) spendiert. Beim Oszilloskop wurde vor allem die Triggerung und die Anzeige statistischer Werte sowie die Kopplung (AC/DC) optimiert. Der mit UDFs erweiterte SWA erlaubt die Generierung einer FFT jeder kontinuierlichen mathematischen Funktion. Ein sehr nützliches Tool!

Einfacherer Einsatz
Hier gab es große Verbesserungen: Nicht nur, dass die Optik jetzt mehr einem modernen Messinstrument entspricht, sondern ganz generell ist die Bedienung nun deutlich intuitiver und konsistenter. Das neue UI (**U**ser **I**nterface) überzeugt schon nach wenigen Minuten.

Die Verbesserungen bezüglich Funktionen und Bedienung ergeben sich schon allein durch die neue Client-Software. Doch für DHCP und höhere Leistung muss man auch die Firmware auf die Version NCSA II upgraden.

Das User Interface

Einen Eindruck vom neuen NCSA-II-UI vermittelt das typische Beispiel in **Bild 1**.

Verbindung mit NCSA II
Beim Start der Client-Software ist die IP-Adresse von NCSA II noch nicht bekannt. Via *„Find“* in der Ecke oben rechts wird nach bis zu zwei NCSAs im LAN gefahndet. Entdeckte IP-Adressen werden in eine Drop-Down-Liste eingetragen. Wenn man dort das gewünschte Gerät auswählt, kann man sich via *„Connect“* damit verbinden.

Sampling
Wie schon bei der alten Version kann man die Sampling-Parameter im Panel *„Sampling“* konfigurieren. Einstellbare Parameter sind Samples/s, Sample-Anzahl, Auflösung (10 oder 12 bit) und der ADC-Takt (ADC-Taktfrequenz-Teiler). Zur automatischen Einstellung der besten Abtastrate bei der Bewegung des entsprechenden Sliders gibt es jetzt die Funktion *„Prevent Jitter“*. Ist sie aktiv, wird automatisch die nächstmögliche Abtastrate mit dem geringsten Jitter eingestellt. Die errechnete Abtastrate ist dann ein ganzzahliges Vielfaches der für eine AD-Wandlung nötigen Zyklen geteilt durch die gewünschte Frequenz.

Trigger
Die Triggerung wird in Software erledigt und die Einstellungen entsprechen einem normalen Oszilloskop. Wird der Trigger-Pegel auf einen bestimmten Wert (z.B. 0,1 V) und die Flanke z.B. auf „ansteigend“ eingestellt, durchsucht ein Algorithmus die vom NCSA II gelieferten Daten und legt eine Liste all der Datenpunkte an, für die gilt: ≥ Trigger-Pegel und beim direkten Vorgänger < Trigger-Pegel (siehe **Tabelle 1**). Diese Liste indiziert dann all die Datenpunkte, die den Trigger-Kriterien entsprechen. Die Anzeige beginnt dann immer beim ersten gefundenen Trigger-Punkt.
Rechts im UI sieht man das Panel *„Trigger“*. Hier kann man einen Trigger-Pegel von -250 bis zu +250 mV und die Trigger-Flanke auf steigend oder fallend einstellen. Außerdem gibt es hier noch die Modi *„Auto“* oder *„Normal“* zur Auswahl. Im normalen Modus wird die Kurve nur dann angezeigt, wenn eine Triggerung erfolgt ist – ohne Triggerung ergibt sich also eine leere Anzeige. Im Auto-Modus wird nach Verstreichen von 200 ms ohne Trigger trotzdem eine Darstellung des Signals ausgelöst, sodass man auch ohne Trigger das Signal sehen kann. Letzteres ist wie bei einem normalen Oszilloskop voreingestellt. Man kann die Triggerung via *„Off“* auch gänzlich abschalten. Den Trigger-Pegel stellt man mit dem Slider oben rechts ein. Via *„Set 0V“* kann man den Trigger-Pegel schnell auf 0 V setzen. Der Signal-Analyzer befindet sich normalerweise im Modus *„Stop“*. Wenn man auf den roten Schalter *„Stop“* klickt, wird in den Modus *„Run“* gewechselt. Die Aufzeichnung kann durch einen weiteren Klick auf *„Run“* beendet werden. Per *„Single“* kann man die Aufzeichnung eines einzelnen Schriebs veranlassen. Hierbei wartet das System auf eine erfolgreiche Triggerung und schaltet dann automatisch wieder in den Modus *„Stop“*.

Tabelle 1. Trigger auf 1 V, ansteigende Flanke. An Position 1039 ergibt sich ein Trigger-Punkt im Daten-Array.

Index	1034	1035	1036	1037	1038	**1039**	1040	1041	1042
Spannung	0,800	0,820	0,883	0,950	0,969	1,003	1,012	1,133	1,306

Koppelung
Im Panel „Coupling“ kann man die Art der Signalkopplung einstellen. Bei *„DC“* wird das Signal so angezeigt, wie es anliegt. Bei *„AC“* wird vom gemessenen Signal via FFT vor der Anzeige der Gleichspannungspegel abgezogen.

Statistik
Aus den vom NCSA-Server gelieferten Messwerten werden diverse statistische Maße wie Minimum, Maximum, Mittelwert und Hub der Spitzenwerte berechnet und angezeigt. Man kann sie per *„Reset“* löschen. Die statistischen Werte findet man oben im UI. Die Aktivierung von *„Draw max trace“* und *„Draw min trace“* erzeugt horizontale Linien mit den maximalen bzw. minimalen Werten zur Zeit der Aktivierung dieser Checkboxen. Die Linien verschwinden via *„Reset“* oder durch Deaktivierung der Checkboxen. Die gleiche Funktion wurde für den Maximalpegel der FFT implementiert.
Unter dem Oszillogramm werden der DC-Offset und die berechnete Signalfrequenz zusammen mit der aktuellen Abtastrate angezeigt. Der DC-Offset wird via FFT und die Frequenz aus der Zeit zwischen zwei Trigger-Punkten errechnet. Beim Frequenz-Spektrum wird zudem auch die Grundfrequenz als Frequenz des höchsten Peaks ermittelt.

Weißes Rauschen kalibrieren
NSCA II kann jetzt auch den Frequenzgang z.B. eines Filters ermitteln. Hierbei kann der eingebaute Rauschgenerator

Eigenschaften

- Netzwerkanschluss
- Plug & Play
- Rauschgenerator
- Oszilloskop
- Spektrum-Analyzer
- Synthetische Signalanalyse
- Subsampling

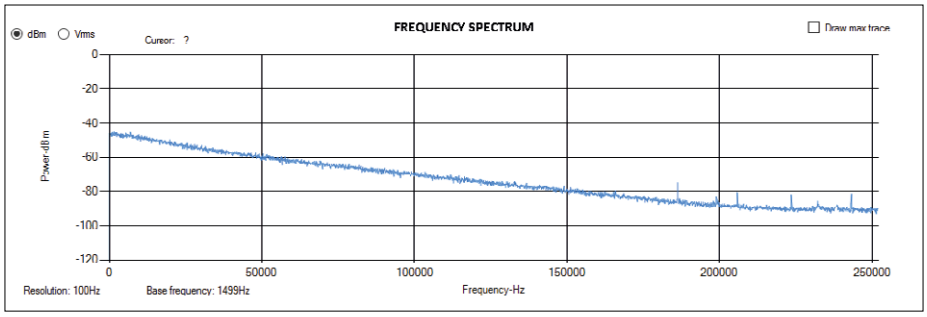


Bild 2. Rausch-Spektrum. Eingang und Ausgang sind verbunden, aber noch nicht kalibriert.

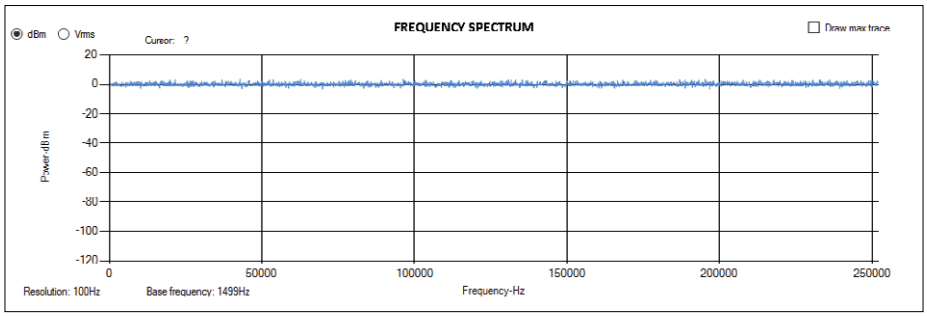


Bild 3. Rauschspektrum von Bild 2 nach Kalibrierung.

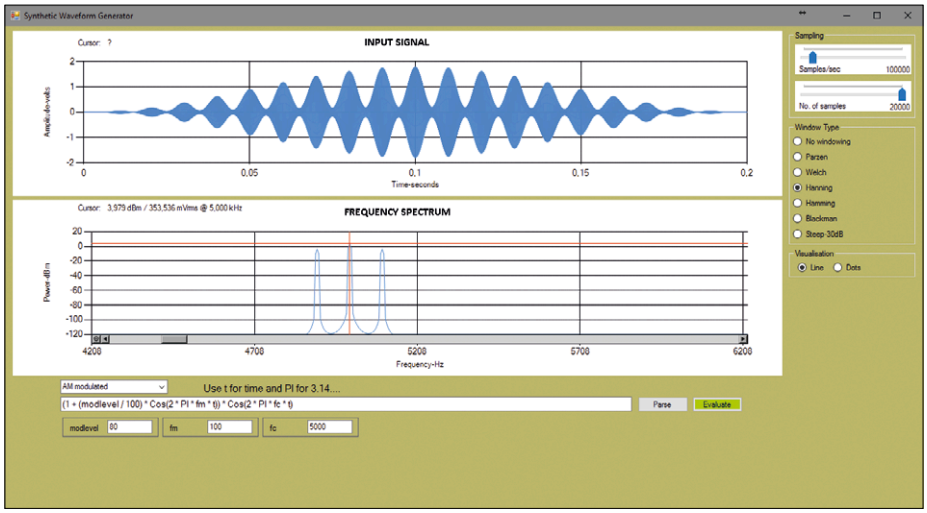


Bild 4. Mit den anwenderspezifischen Fähigkeiten des SWA kann man ein amplitudenmoduliertes Signal generieren und analysieren. Die Parameter für Modulationstiefe, Trägerfrequenz und Modulationsfrequenz können in die entsprechenden Felder eingegeben werden.

Tabelle 2. Stellungen von JP2 und IP-Adressen.

RB13 (JP2 Pin 3)	RB14 (JP2 Pin 1)	IP-Adresse
Masse	Masse	192.168.1.123 fix
Masse	offen	192.168.0.123 fix
offen	Masse	DHCP 1
offen	offen	DHCP 2

als Signalquelle für das untersuchte Filter verwendet werden. Weißes Rauschen sollte alle Frequenzen mit annähernd gleichem Pegel enthalten. Je nach Frequenzgang werden die Pegel bestimmter Frequenzen vom Filter abgeschwächt oder verstärkt. Sein Ausgangssignal wird gemessen und angezeigt.

Da der Ausgang des Rauschgenerators mit einem Tiefpassfilter versehen ist, ist der Rauschpegel leider nicht für alle Frequenzen gleich. Mit einer Kalibration lässt sich diese Frequenzgangverzerrung kompensieren. Hierzu wird das erzeugte Rauschen direkt vom NCSA II gemessen. Dabei wird die maximale Anzahl an FFT-Durchläufen zur Mittelung (= 500) verwendet. Nach diesen 500 Durchläufen wird das generierte Spektrum angezeigt. Klickt man dann auf *„Calibrate“*, wird dieses Spektrum zur Kalibrierung genutzt, auch für zukünftige Messungen (siehe **Bilder 2** und **3**). Nach der Kalibrierung kann man ein Filter zwischen Rauschausgang und Messeingang legen und dessen korrigierter und somit realer Frequenzgang zeigt sich im Spektrum.

Die Einstellungen für eine Kalibrierung können durch *„Auto set“* automatisch vorgenommen werden. Ein wichtiger Hinweis: Wenn der Eingang der zu untersuchenden Schaltung hochohmig ist und eben keine Impedanz von 50 Ω aufweist, sollte man einen 50-Ω-Widerstand parallel zum Eingang der Schaltung legen. Durch diesen Trick wird der Rauschausgang genauso mit 50 Ω belastet, wie wenn er direkt mit dem Messeingang des NCSA II verbunden wäre. Da der NCSA-II-Eingang gleichspannungsgekoppelt ist, sollte die untersuchte Schaltung gegebenenfalls am Ausgang mit einem (ausreichend großen) Serienkondensator versehen sein, um eventuelle Gleichspannungen abzutrennen.

SWA

Die synthetische Kurvenform-Analyse wurde ebenfalls erweitert. Jetzt können eigene Formeln bezüglich Abtastfrequenz, Sample-Anzahl und FFT-Fenster eingegeben und angewendet werden.

Der SWA nutzt die Open-Source-Library NCalc für das Parsen der eingegebenen Formel. Nach Eingabe der Formel und Klick auf *„Parse“* werden die noch unbekannten Variablen als Eingabefelder unterhalb der Zeile mit der Formel angezeigt. Hier kann man dann die entsprechenden Werte eingeben (**Bild 4**). Nach Klick auf *„Evaluate“* werden diese

Werte übernommen und die Formel für alle Samples angewendet. Das Ergebnis wird dann im Zeitbereich oben angezeigt. Außerdem wird auch ein Spektrum berechnet und darunter dargestellt.

PnP-Netzwerk

Zum Netzanschluss konnte man bei der alten Version mit JP2 eine von vier fixen IP-Adressen einstellen. Die neue Firmware spricht aber auch DHCP, weshalb sie den DHCP-Server automatisch um eine IP-Adresse bittet, sobald NCSA II an ein LAN angeschlossen wird. Außerdem ist es jetzt möglich, gleichzeitig zwei NCSA II in einem LAN kollisionsfrei zu betreiben.

DHCP-Adresszuweisung

Im Prinzip soll lediglich eine Verbindung von einem PC als Client zu einem NCSA II als Server hergestellt werden. Hierzu muss der Client die IP-Adresse des Servers kennen. DHCP vereinfacht die Adresszuweisung an den Server. Praktisch jedes LAN verfügt über einen DHCP-Server (normalerweise im heimischen DSL- oder Kabel-Router), der auf Anfrage eine IP-Adresse herausgibt. Die einzelnen DHCP-Phasen kann man mit den Debug-Fähigkeiten des NCSA II untersuchen. Debugging ist möglich, wenn man einen via USB an einen PC angeschlossenen USB/Seriell-Konverter mit dem NCSA-Debug-Anschluss K6 verbindet. Mit Hilfe von Terminal-Software (z.B. Tera Term) erhält man dann Einblick in die verschiedenen Stufen der DHCP-Kommunikation (**Bild 5**).

Mit JP2 ergibt sich eine Auswahl zwischen vier IP-Adressen (siehe **Tabelle 2**). Bei den ersten beiden handelt es sich um fixe Adressen wie beim alten NCSA ohne „II“. Die beiden restlichen Einträge sind für zwei via DHCP bezogene IP-Adressen gedacht. Hierdurch können also zwei NCSAs im gleichen LAN betrieben werden.

Mit DHCP erhält NCSA II zwar eine IP-Adresse vom DHCP-Server, doch der Client auf dem PC bekommt davon zunächst einmal nichts mit. Wie erfährt also der Client die IP-Adresse eines NCSA II? Das Problem ist komplexer, als man annehmen sollte.

Suche nach NCSA II

Die Datei *NetworkScanner.cs* im C#-Projekt enthält den Code, der dem PC das Auffinden eines NCSA II bzw. seiner IP-Adresse ermöglicht. Nachfolgend eine kurze Führung durch die einzelnen Schritte, garniert mit einzelnen Code-Schnipseln von *NetworkScanner.cs*:

1. (**Listing 1**) Finde die IP-Adresse des LAN-Gateways (normalerweise der Router) und prüfe, ob es aktiv ist und läuft. Falls das zutrifft, wird die Gateway-IP-Adresse geholt. Eine typische Gateway- bzw. Router-Adresse wäre 192.168.0.1.
2. (**Listing 2**) Schaue im lokalen Adressbereich des Gateways nach einem NCSA II mit IP-Adresse. Hierzu werden alle Adressen im lokalen Bereich angepingt. Falls die Gateway-Adresse 192.168.0.1 ist, erfolgen Pings an die Adressen 192.168.0.2 bis 192.168.0.255. Dies würde ziemlich dauern, wenn man eine Adresse nach der anderen abfragen würde. Daher werden parallele Abfragen via Multithreading gestartet. Wenn ein Ping abgeschlossen ist, wird die damit verbundene Methode *PingCompleted* ausgeführt.
3. (**Listing 3**) Bei einem abgeschlossenem Ping wird geprüft, ob die Abfrage erfolgreich war. Falls ja, wird versucht, eine TCP/IP-Verbindung zu dieser Adresse via Port 4000 (hart codiert

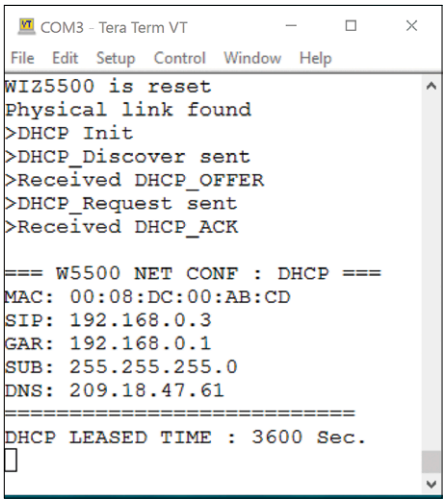


Bild 5. Dank NCSA-Debug-Ausgang kann man die DHCP-Phasen inspizieren.

Listing 2. Suche nach NCSA-Geräten.

```
private static void Ping(IPAddress host, int attempts, int timeout)
{
  ...
  Ping ping = new Ping();
  ping.PingCompleted += PingCompleted;
  ping.SendAsync(host, timeout, host);
  ...
}
```

Listing 1. Ermittlung der IP-Adresse des Gateways.

```
static IPAddress NetworkGateway()
{
  ...
  foreach (NetworkInterface f in NetworkInterface.GetAllNetworkInterfaces())
  {
    if (f.OperationalStatus == OperationalStatus.Up)
    {
      foreach (GatewayIPAddressInformation d in f.GetIPProperties().GatewayAddresses)
      {
        if (d.Address.AddressFamily == AddressFamily.InterNetwork)
          ip = d.Address;
}  }  }  }
```

im TCP/IP-Code von NCSA II) aufzubauen. Wenn diese Verbindung erfolgreich aufgebaut werden kann, wird die MAC-Adresse (**M**edia **A**ccess **C**ontrol) des NCCSA II angefordert.

4. (**Listing 4**) Um diese MAC-Adresse zu erhalten, muss man die ARP-Tabelle (**A**ddress **R**esolution **P**rotocol) durchsuchen. Diese Tabelle listet die zu den IP-Adressen gehörenden MAC-Adressen von Geräten auf, die mit diesem LAN verbunden sind.

Bei jeder erfolgreichen TCP/IP-Verbindung ergänzt der PC die ARP-Tabelle um das jeweilige Paar an IP/MAC-Adressen. Anschließend findet man in dieser Tabelle die MAC-Adressen eines NCSA II (oder deren zwei, siehe **Bild 6**).
Dank unterschiedlicher MAC-Adressen ist es möglich, zwei NCSA II zugleich in einem LAN zu betreiben. Die NCSA-Firmware NCSA ändert nämlich den signifikanten Teil der Standard-MAC-Adresse eines W5500 anhand der Position von JP2. Findet der PC also zwei unterschiedliche NCSA-MAC-Adressen, ist klar, dass auch zwei unterschiedliche IP-Adressen vorhanden sind. Tatsächlich ließe sich sogar ein dritter NCSA II im gleichen Netz betreiben, indem das dritte Gerät eine fixe IP-Adresse erhält.
Das war's auch schon. Wenn ein NCSA II über eine gültige IP-Adresse verfügt, und diese dem PC bekannt ist, kann auch eine TCP/IP-Verbindung aufgebaut werden, womit sich Daten übertragen lassen. Dieses Verfahren erleichtert die Einrichtung eines NCSA II und den Netzbetrieb beträchtlich.

Tuning

Die Verbesserungen bei der Arbeitsgeschwindigkeit eines NCSA II sind hauptsächlich eine Folge von Multithreading bei der Client-Software. Hinzu kommt noch der Einsatz von DMA beim (auf einem dsPIC33 basierenden) Server. Nachfolgend eine kurze Beschreibung:

Multithreading

Bei einem Programm ohne Multithreading gibt es im Prinzip einen einzigen Thread, dessen Befehle nacheinander abgearbeitet werden. Dieser Haupt-Thread wird beim Start des Programms durch das OS generiert. Der eine Thread wird auch nur auf einem einzigen Core der CPU ausgeführt. Bei einem PC mit Quad-Core-CPU werden daher nur maximal 25 % dessen Potentials ausgenutzt. Erst Multithreading nutzt das Potential einer Multi-Core-CPU richtig aus.
Bei der ersten Version der Client-Software wurde alles in einem Thread nacheinander ausgeführt: Datenanforderung vom NCSA-Server, Empfang der angeforderten Daten, Umwandlung von Volt in dB, FFT-Berechnung, Anzeige des Oszillogramms und des Spektrums. Wenn eine oder mehrere Teilfunktionen mehr Zeit benötigen, reduziert das direkt die Update-Rate des Bildschirms. Im einzigen Thread muss ja noch zusätzlich das UI bedient werden. Wenn daher gerade andere Dinge erledigt werden, reagiert die Oberfläche recht träge auf Eingaben wie etwa einen Tastendruck. Auch das Update des Bildschirms selbst braucht Rechenzeit. Die Lösung für eine Beschleunigung ist daher, jede wichtige Funktion in einen eigenen Thread zu packen, was die Last auf mehr Kerne der CPU verteilt.
In der neuen Client-Software werden Anforderung und Übertragung von Daten in separaten Threads erledigt. Auch die FFT-Berechnungen erfolgen in einem eigenen Thread. Dadurch kann mehr parallel erledigt werden, was die Leistung massiv steigert. Die Update-Rate der Anzeige steigt und das UI reagiert viel unmittelbarer. Während die Anzeigen vom Haupt-Thread gerendert werden, holen andere Threads schon die nächsten Daten und berechnen die zugehö-

Tabelle 3. Unterschiede in den Update-Raten mit und ohne Optimierungen (f_s = 504.201 Hz).

N	NCSA Update-Rate (Hz)	NCSA II Update-Rate (Hz)
5.000	8,33	42
10.000	3,96	22
15.000	2,48	14
20.000	1,70	11

Listing 3. Antwortet ein NCSA?

```
private static void PingCompleted(object sender,
PingCompletedEventArgs e)
{
  ...
  if (e.Reply != null && e.Reply.Status == IPStatus.Success)
  {
    // Try to connect to the port
    if (TryConnect(ip, _port))
    {
      // Can connect.
      string macaddres = GetMacAddress(ip);
      ...
}  }  }
```

Listing 4. Mit ARP-Tabelle die IP- zur MAC-Adresse finden.

```
public static string GetMacAddress(IPAddress ip)
{
  ...
  if (SendARP(BitConverter.ToInt32(ip.GetAddressBytes(),0),0,macAddr,ref macAddrLen) != 0)
    return "00:00:00:00:00:00";
  ...
}
```

rigen FFTs. Beim selben PC stieg hier die Update-Rate des Bildschirms von 8 Hz auf immerhin 40 Hz bei der gleichen Datenmenge von 5.042 Samples bei 504.021 S/s (**Tabelle 3**). Dies entspricht einer Steigerung um den Faktor 5!

DMA

In der alten Version der NCSA-Firmware wird die dsPIC33-SPI-Peripherie mit Hilfe der Funktionen `wiznet_read()` und `wiznet_write()` aus der I/O-Library *W5500.c* angesprochen, um Daten mit Hilfe des Netzwerk-Chips W5500 zu transferieren. Die Übertragungsgeschwindigkeit wird dabei durch die Anzahl der von den Library-Funktionen benötigten Instruktionen begrenzt, die ja von der dsPIC-MCU ausgeführt werden müssen. Die neue Firmware nutzt aber die DMA-Fähigkeiten des dsPIC33, was das Tempo massiv erhöht, da der Transport der Daten von der SPI-Peripherie in den Speicher direkt ohne Beteiligung der CPU innerhalb der MCU erfolgt. Die Datenrate hat sich dabei von 700 KB/s auf 1,3 MB/s fast verdoppelt.

Leider ist die Verwendung von DMA bei SPI-Transfers bei der MCU dsPIC33 nicht ganz trivial. Man braucht zwei DMA-Kanäle, damit das funktioniert. Kanal 1 schickt Bytes an den W5500 und DMA-Kanal 2 muss nach jedem Byte ein „dummy read" ausführen, da dies vom DMA-System des dsPIC33 verlangt wird. Beide DMA-Kanäle transferieren einen Datenblock und lösen anschließend Interrupts aus. Dann warten die DMA-Kanäle, bis die nächste Datenanforderung eintrifft. **Listing 5** zeigt, wie die DMA-Steuerung im Zusammenhang mit SPI funktioniert.

Ein „dummy read" nach jedem Byte ist ziemlich nichtideal. Auch dass man manuell den Transfer des ersten Bytes einer Folge erzwingen muss, ist suboptimal. Viel einfacher wäre es, einen DMA-Kanal zu requirieren, den Datentransfer zu starten und zu warten, bis er fertig ist. Leider ist das beim dsPIC33 nicht so simpel implementiert.

NCSA-Upgrade

Ein alter NCSA I funktioniert auch mit der neuen Client-Software. Damit hat man schon einmal das neue UI mit verbessertem SWA. Das Anfordern der IP-Adresse per DHCP und Geschwindigkeitsverbesserungen beim Server bleiben aber der neuen Firmware und damit dem NCSA II vorbehalten.

Listing 5. SPI mit DMA beim dsPIC33.

```
done = false;  //using dma and spi for faster uP to W5500 transfers
SPI1STATbits.SPIEN=0; //keep SPI off until ready for transfer

//Set up DMA 1 to transfer from RAM at pBuf to SPI
DMA1CNT = len-1;  //block size
DMA1STAL = (unsigned int) pBuf;  //point to data buffer
IFS0bits.DMA1IF = 0;  //clear DMA interrupt flag
IEC0bits.DMA1IE = 1;  // Enable DMA interrupt
DMA1CONbits.CHEN = 1;  //turn DMA1 on

//Set up DMA 2 to do dummy reads after each write;
//required for dsPIC33 DMA SPI
DMA2STAL = (unsigned int) &dummy;
DMA2CNT = len-1;
IFS1bits.DMA2IF = 0;
IEC1bits.DMA2IE = 1;
DMA2CONbits.CHEN=1;

//force the first byte transfer after enabling SPI;
//required by dsPIC33 to start process
SPI1STATbits.SPIEN=1;//turn on SPI
DMA1REQbits.FORCE=1;
while (DMA1REQbits.FORCE==1);

//wait for DMA transfer to complete; done set true by DMA interrupt
while (done==false);
```

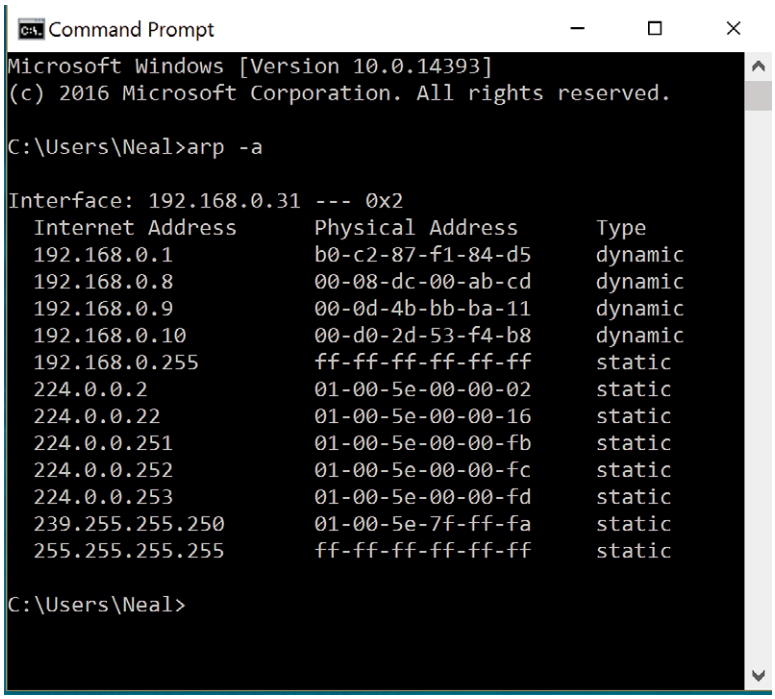


Bild 6. Die ARP-Tabelle verrät einen NCSA im LAN (Eingabe von `arp -a` nach einem Befehls-Prompt zeigt diese Tabelle). Die IP-Adresse 192.168.0.8 gehört zu einem NCSA mit der Standard-MAC Adresse 00-08-dc-00-ab-cd.

Frequenz mit kalibriertem Rauschen

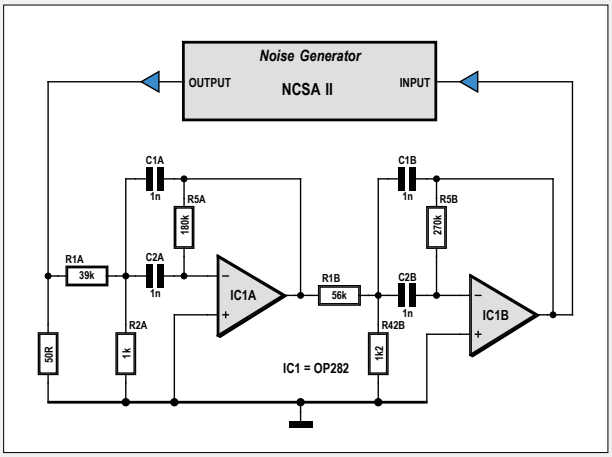


Bild 7. Analyse eines aktiven Bandpass-Filters mit Hilfe des im NCSA eingebauten Rauschgenerators.

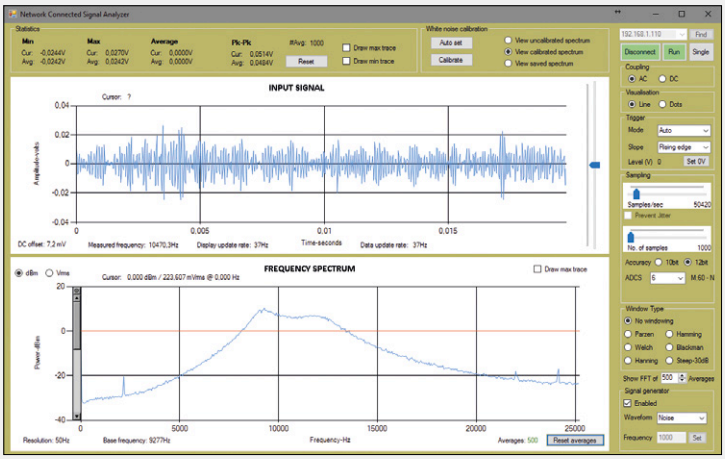


Bild 8. Der resultierende Frequenzgang der Schaltung in Bild 7, erstellt mit kalibriertem Rauschen.

Mit Hilfe des im NCSA II eingebauten Rauschgenerators kann man den Frequenzgang eines aktiven Bandpassfilters wie in **Bild 7** darstellen. Zur Kalibrierung des Frequenzgangs des Rauschens wird dieses zunächst vom NCSA II aufgezeichnet und sein Spektrum als Referenz genutzt. Wird dann der reale Bandpass zwischen Rauschausgang und NCSA-Eingang geschaltet, ergibt sich der kalibrierte Frequenzgang des Filters nach **Bild 8**. Betrachtet man das resultierende Spektrum, erkennt man eine Mittenfrequenz von etwa 10 kHz mit einer Bandbreite von knapp 5 kHz sowie eine Flankensteilheit von annähernd 24 dB/Oktave, wie man dies für ein vierpoliges Bandpass-Filter auch erwarten würde.

Subsampling

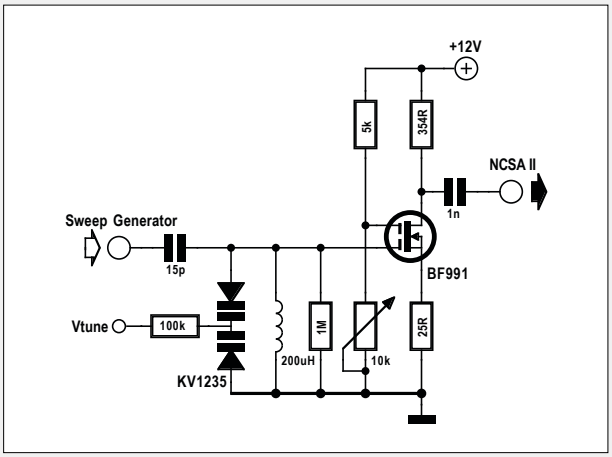


Bild 9. Abstimmbares Mittelwellenradio mit FET-basiertem LC-Bandpass.

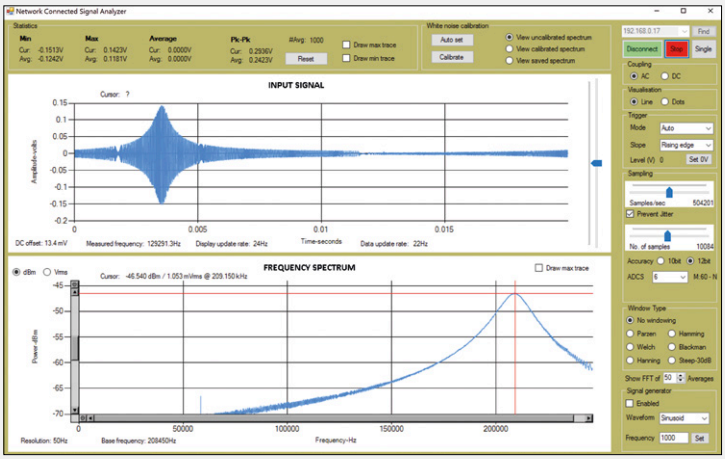


Bild 10. Frequenzgang des HF-Bandpasses, ermittelt mit HF-Sweep-Signalen. Die Mittenfrequenz beträgt 1,201 MHz. Für diese HF-Signale wurde Subsampling genutzt.

Hier wird ein HF-Bandpassfilter mit einer Mittenfrequenz von etwa 1,2 MHz untersucht (**Bild 9**). Es handelt sich dabei um einen per FET abgestimmten LC-Bandpass, der als Eingangsstufe für ein Mittelwellenradio dient. Die Mittenfrequenz kann über die Spannung *Vtune* verändert werden, da sie die Kapazität zweier Kapazitätsdioden verändert, die parallel zu einer Spule geschaltet sind. Der Abstimmbereich reicht von 580 kHz bis 1,6 MHz. Da hierfür die Abtastrate des NCSA laut Nyquist-Kriterium (mindestens die doppelte Abtastfrequenz gegenüber dem Nutzsignal) nicht ausreicht, kommt hier der NCSA-Subsampling-Modus [2] zum Einsatz.

Mit Hilfe eines externen Signalgenerators wird ein Sweep-HF-Nutzsignal an den Eingang des Filters gelegt (Sweeps von 1...1,5 MHz in 10 ms). Das Filter ist auf etwa 1,2 MHz abgestimmt. **Bild 10** zeigt den Frequenzgang des Filters. Im Zeitbereich oben sieht man den Pegel am Filterausgang über der Zeit. Im Spektrum darunter erkennt man einen Peak bei 209.150 Hz. Aufgrund des Subsamplings muss man hier ein Vielfaches der Abtastfrequenz addieren. Mit gemessenen 209.150 Hz + 2 × 504.201 Hz (Abtastfrequenz) ergibt sich die reale Mittenfrequenz f_c = 1.217.552 Hz.

QAM-Modulator

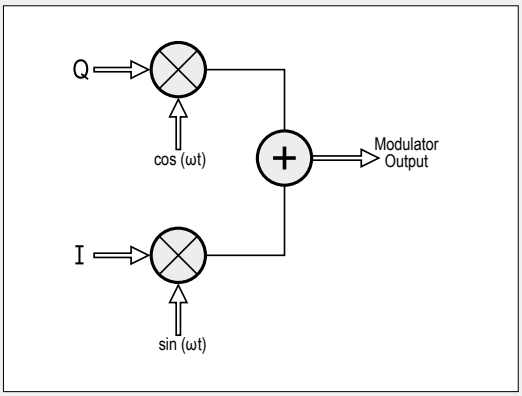


Bild 11. Blockschaltung eines IQ-Modulators.

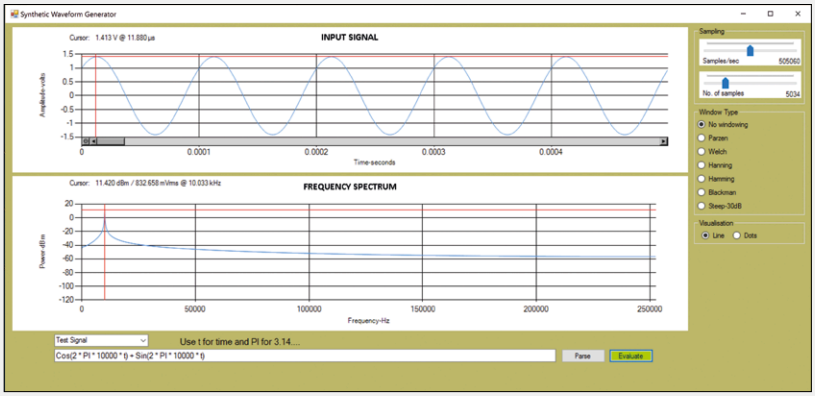


Bild 12. Der IQ-Modulator-Ausgang des NCSA-SWA.

Der NCSA-SWA bietet UDF-Möglichkeiten. In diesem Beispiel wird damit ein klassischer QAM- bzw. IQ-Modulator realisiert und sein Verhalten dargestellt. Diese Modulationsform ist aus moderner Kommunikationstechnik nicht wegzudenken. **Bild 11** zeigt eine typische IQ-Modulator-Konfiguration. Beim Kabelfernsehen wird vielfach eine 256-QAM verwendet, wobei I und Q dann jeweils 16 unterschiedliche Amplituden annehmen können. Die Gleichung für den IQ-Modulator ist:

$$Q \cos(\omega t) + I \sin(\omega t) = C \cos(\omega t + \theta)$$

wobei gilt: $C = \sqrt{(Q^2 + I^2)}$ und $\theta = \tan^{-1}(Q / I)$

Laut der Gleichung ergibt sich für bestimmte I- und Q-Werte am Modulator-Ausgang eine Cosinus-Welle, mit der durch die Gleichung angegebenen Amplitude und Phase. Wenn man diese Funktion beim NCSA II eingibt, kann man überprüfen, ob Software und Hardware mit der Theorie übereinstimmen.

In diesem Beispiel gilt I = Q = 1 V. Laut der Gleichung sollte sich ein Cosinus mit einer Anfangs-Phase bei t = 0 von 45 ° ergeben, und die Amplitude sollte 1,414 V sein. **Bild 12** beweist, dass die sich ergebenden Kurven den theoretischen Werten entsprechen. Man kann in den Eingabefeldern die Werte für I und Q ändern und die resultierenden Änderungen betrachten.

Für den neuen Client-Code braucht man zunächst Visual Studio 2015, das kostenlos auf den Servern von Microsoft zum Download zur Verfügung steht. Der zweite Download betrifft den Code: Das zugehörige Visual Studio NCSA-Client-Projekt wartet unter [3] auf den Download. Nach dem Projekt-Build ist das Upgrade fertig.
Für das Update der Firmware auf Version II benötigt man die IDE MPLABS, die es kostenlos von Microchip gibt. Außerdem wird natürlich noch das MPLABS.X-NCSA-Projekt von [3] benötigt. Nachdem der Code kompiliert wurde, schiebt man die neue Firmware mit einem Programmer in den dsPIC33. Hierzu tut es der preiswerte und direkt auf den ICSP-Header K1 steckbare PICkit-3-Programmer von Microchip.
Allen NCSA-I-Besitzern, die keinen Programmer zur Verfügung haben, bietet Elektor auch eine Neuprogrammierung des Controllers an.

Was noch?

Es bleiben immer noch viele Möglichkeiten, NCSA II mit neuen Features auszustatten. Mit Double-Buffering könnte man das Tempo weiter beschleunigen, da so keine Lücken im Datenfluss mehr entstehen. Damit wären Oszilloskop und FFT quasi echtzeitfähig. Und wer will, kann zusätzlich noch Funktionen eines SDR (**S**oftware **D**efined **R**adio) einbauen.
Auch die anwenderspezifischen SWA-Funktionen ließen sich z.B. mit nichtlinearen Funktionen aufpeppen. Mit Funktionen wie Rect, Sign, Delta etc. ergäben sich gute Analysemöglichkeiten für Kommunikationsprozesse. Doch auch so ist das NCSA II ein mächtiges Analyse-Tool! ⏮

(160362)

Weblinks

[1] NCSA Teil 1: www.elektormagazine.de/150211

[2] NCSA Teil 2: www.elektormagazine.de/150694

[3] www.elektormagazine.de/160362

EINKAUFSZETTEL

→ 150211-91
Fertiges NCSA-Modul

→ 150211-71
Gehäuse für ein NCSA-Modul

→ 160362-41
Neuprogrammierung NCSA nach NCSA II

eLABS PROJECT

4-Kanal-Fernsteuerung

Von **Somnath Bera** (Indien) &
Roy Aarts (Elektor-Labor)

In zwei Varianten: XBee oder nRF24

Eigenschaften

- Vier Kanäle
- Zwei Varianten
- Hohe Reichweite
- Niedrige Kosten

Mit preisgünstigen ISM-Funkmodulen lassen sich leicht einfache ferngesteuerte Spielsachen und andere RC-Geräte bauen, die nicht mehr als einige Schalter und ein paar Motoren besitzen. Geht es jedoch um kompliziertere Dinge wie die Steuerung mehrerer Ruder-Servos eines Flugmodells plus eines bürstenlosen Motors bei gleichzeitiger Rückkontrolle jeder Aktion, dann muss ein zuverlässigeres Funk-System her.

Die XBee-Module von Digi International sind einfach zu bedienende drahtlose Transceiver-Module. Sie wurden ursprünglich für die Punkt-zu-Punkt- und Stern-Kommunikation mit einer Übertragungsgeschwindigkeit von 250 kbit/s entwickelt; es gibt Typen für den Mobilfunk, ZigBee, Thread, 802.15.4 und WLAN.

In diesem Artikel verwenden wir XBee-S2C-Module (vormals ZB) mit einem ZigBee-Protokollstack.

XBee-Module sind zwar schön einfach zu bedienen, können aber auch recht teuer sein. Deshalb hat das Elektor-Labor zusätzlich eine ganz ähnliche Steuerung entworfen, die mit den günstigen nRF24-Funkmodulen auskommt. So behandelt dieser Artikel gleich zwei Vier-Kanal-Fernbedienungen, eine mit XBee- und eine mit nRF24-Modulen. Beginnen wir mit der ersten.

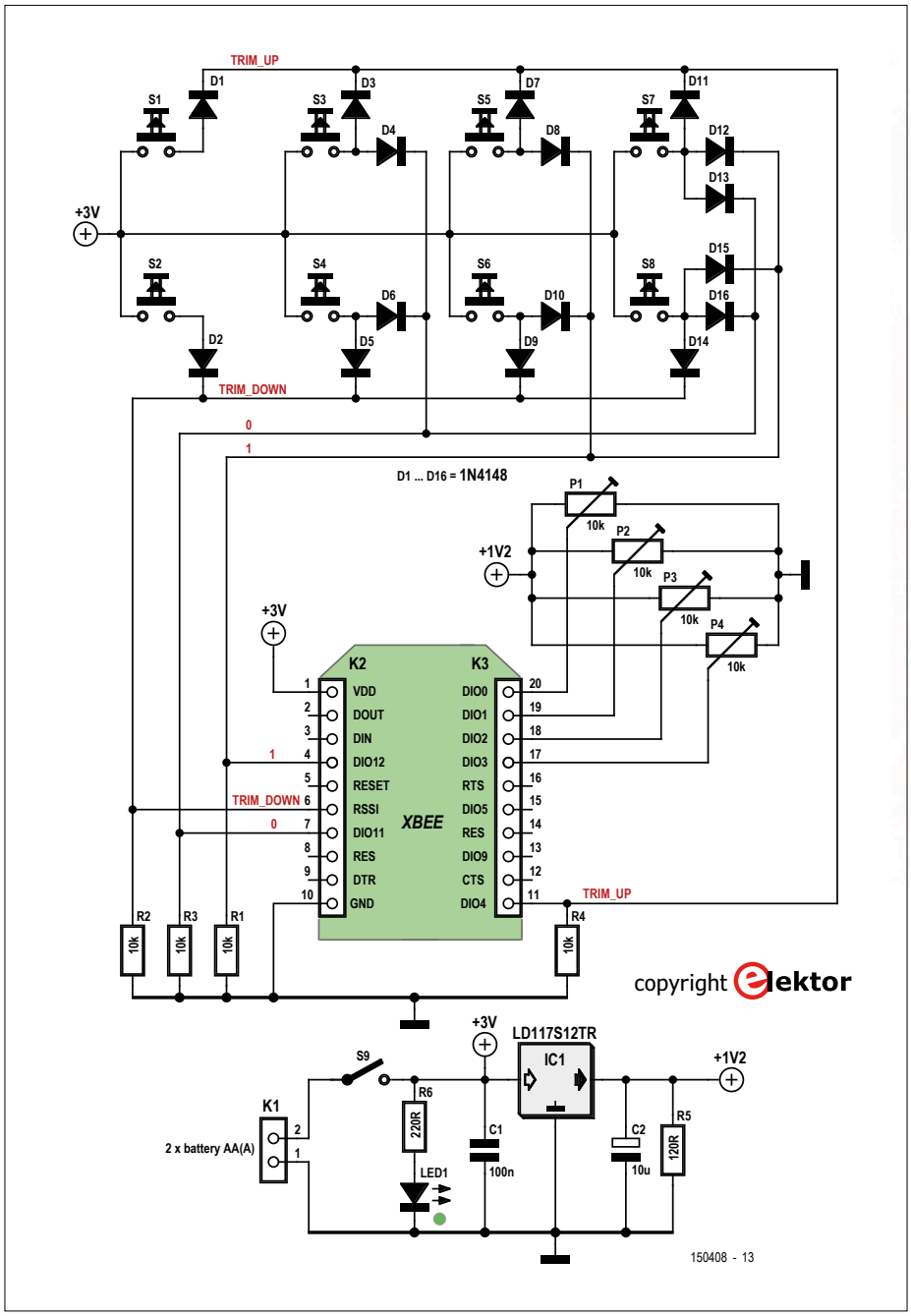
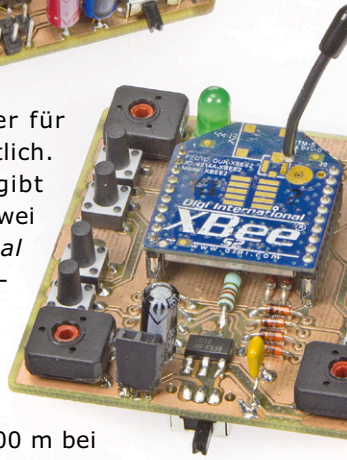


Bild 1. Schaltung des XBee-Senders.

XBee

XBee-Funkmodule gibt es in vielen Ausführungen, die mehrere Netzwerk-Topologien unterstützen. Für jeden Netzwerktyp sind die Module mit unterschiedlichen Antennenkonfigurationen und Befestigungsmöglichkeiten (SMD oder für Stiftleiste) erhältlich. ZigBee-Module gibt es außerdem in zwei Versionen: *normal* mit einer Reichweite von 60 m in Innenräumen und enger städtischer Umgebung und von etwa 1200 m bei freier Sicht (line-of-sight, LOS) sowie *Pro* mit einer Reichweite von 90 m beziehungsweise 3,2 km LOS. Mit einer Yagi-Richtantenne kann die Pro-Version eine LOS-Reichweite von bis zu 20 km erzielen! Beachten Sie, dass der XBee-Pro leider nicht für den Einsatz in Europa zertifiziert ist.



Die Module sind in etwa so groß wie eine 2-€-Münze, allerdings mit 24 × 28 mm² quadratisch, wobei die Pro- einen Tick länger ist als die Normal-Version. Die Pro-Version liegt auch bei der Leistungsaufnahme knapp in Front. Die Module verfügen über zwei 1x10-polige Stiftleisten, die 15 digitale I/O-Leitungen und/

oder vier analoge Eingänge zur Verfügung stellen. Wie erwähnt, sind mehrere Antennenkonfigurationen möglich: Bei den Through-hole-Modulen gibt es Varianten für Chipantennen, Draht, U.FL und RP-SMA. Wir haben uns für den einfachen Draht-Antennen-Typ entschieden, der gut genug ist, um damit alle möglichen XBee-Experimente anstellen zu können. Die ZigBee-Module sind zwar für Mesh-Netzwerke und Router gedacht und es können mit anderen Netzwerkgeräten so auch große Netze mit großer Reichweite geschaffen werden, aber auch eine einfache Punkt-zu-Punkt-Verbindung, wie wir sie anstreben, ist möglich.

Der XBee-Sender

Bild 1 zeigt die Schaltung des XBee-Senders. Da das Modul einen vorprogrammierten 32-Bit-ARM-Cortex-M3-Prozessor besitzt, kann es alleine und ohne Unterstützung eines externen Mikrocontrollers arbeiten. Die XBee-Firmware scannt alle analogen und digitalen Eingänge und überträgt die Werte über eine virtuelle (drahtlose) serielle Verbindung an einen Empfänger.

Vier der I/O-Anschlüsse des Moduls werden hier verwendet, um Spannungen zu lesen. Wir haben Trimmpotis eingesetzt, aber wenn Sie ein Flugzeug oder ein Auto steuern möchten, ist es besser, sie durch zwei Joysticks zu ersetzen. Die horizontale Achse jedes Joysticks kann anstelle von P1/P3 angeschlossen werden, die vertikale Achse ersetzt P2/P4. Man kann ein Trimmpoti auch durch einen LDR oder einen anderen Sensor ersetzen, der wie ein Spannungsteiler funktioniert. Da die Analogeingänge Spannungen bis zu 1,2 V lesen können, werden die Trimmpotis/Sensoren auch vom 1,2-V-Spannungsregler IC1 (unabhängig von der Batteriespannung) versorgt.

Mit dem XBee-Sender sollen Servos gesteuert werden, die manchmal einen kleinen konstanten Versatz (Offset) benötigen, um eine mechanische Ungenauigkeit auszugleichen. Dies geschieht mit den Tastern S1...S8. Die ungeradzahligen Drucktasten erhöhen diesen Offset, die geradzahligen verringern ihn. Die Drucktasten sind so wie in **Tabelle 1** beschrieben den Kanälen zugeordnet. Die 16 Dioden D1...D16 kodieren die neun Möglichkeiten (auch „keine Taste gedrückt“ ist eine gültige und wichtige Position) binär auf drei I/O-Leitungen. Man sollte aber nicht mehrere Tasten gleichzeitig drücken.

Dies war es auch schon mit der Hardware des Senders – zur Software kommen wir später. Gute Nachricht bis dahin: Es muss keine Programmiersprache erlernt werden!

Der XBee-Empfänger

Die Schaltung des Empfängers in **Bild 2** unterscheidet sich gravierend von der des Senders, da nun ein externer Mikrocontroller IC3, ein ATmega328P, zum Dekodieren der vom Sender empfangenen seriellen Daten benötigt wird. Verständlich, denn die Standard-Firmware des

INFOS ZUM PROJEKT

Modellbau
XBee | nRF24 | Arduino
Fernsteuerung | RC

Einsteiger
Fortgeschrittener
Experte

Etwa 2 Stunden.

Computer, Lötkolben, USB-to-Serial-Adapter, AVR-ISP-Programmer

XBee-Version: etwa 100 €
nRF24-Version: etwa 30 €

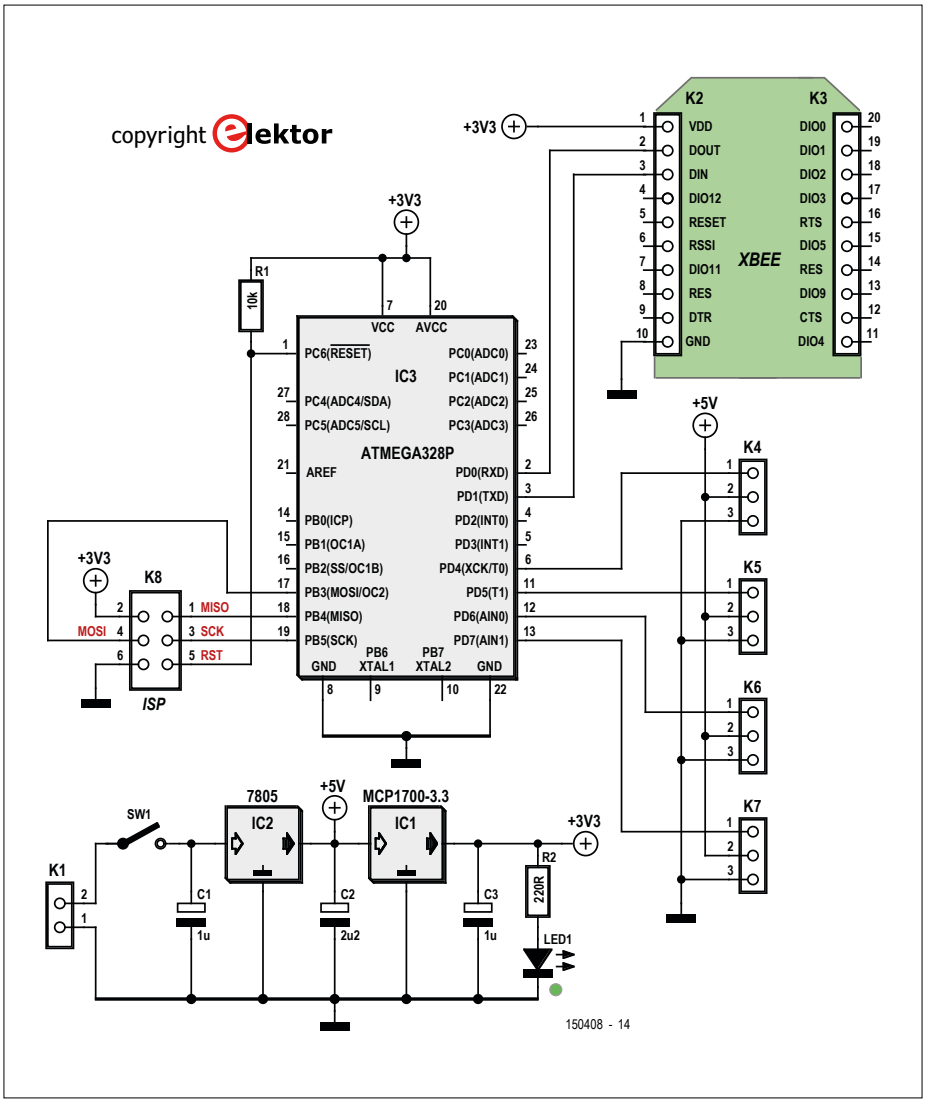


Bild 2. Der Mikrocontroller des XBee-Empfängers dekodiert die empfangenen Daten.

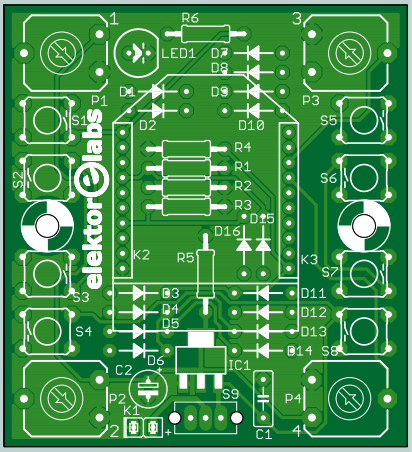
STÜCKLISTE
150408-3 XBee-SENDER

Widerstände:
P1...P4 = 10 k Trimmpoti
R1...4 = 10 k
R5 = 120 Ω
R6 = 220 Ω

Kondensatoren:
C1 = 100 n
C2 = 10 µ, 6,3 V, Raster 2,5 mm

Halbleiter:
IC1 = LD1117S12TR
D1-16 = 1N4148
LED1 = grün, 3 mm

Außerdem:
K1 = 2-polige Stiftleiste, Raster 0,1"
K2,K3 = 1x10-polige Stiftleiste, Raster 2 mm
S1...S8 = Taster, 6x6 mm
S9 = Schiebeschalter (OS102011MS2QN1C)
XBee S2C oder XBee-Pro S2C (+ Antenne)



Platine 150408-3
Keine Firmware

Mikrocontroller programmieren

Alle drei in diesem Artikel verwendeten AVR-Mikrocontroller werden von ihren internen RC-Oszillatoren mit 8 MHz getaktet. Die Konfigurationseinstellungen sind für alle Versionen gleich (siehe Tabelle 3).
Um die Software für diese Controller in der Arduino-IDE zu kompilieren, muss man im Menü „Tools" das richtige „Board" auswählen, zum Beispiel den „Arduino Pro or Pro Mini" und nach dieser Auswahl unter „Processor" die Option „ATmega328 (3.3 V, 8 MHz)". Sie können auch „Atmega328 on a breadboard (8-MHz internal clock)" wie unter „www.arduino.cc/de/Tutorial/ArduinoToBreadboard" beschrieben wählen. Für eine erfolgreiche Kompilierung der nRF24-Software muss außerdem auch die Arduino-RF24-Bibliothek (https://github.com/TMRh20/RF24) installiert werden.

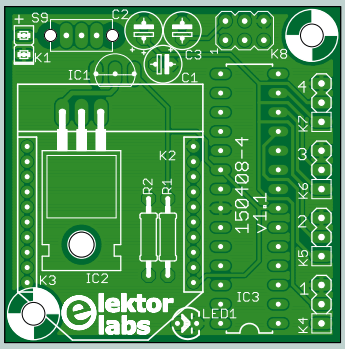
STÜCKLISTE
150408-4 XBee-EMPFÄNGER

Widerstände:
R1 = 10 k
R2 = 220 Ω

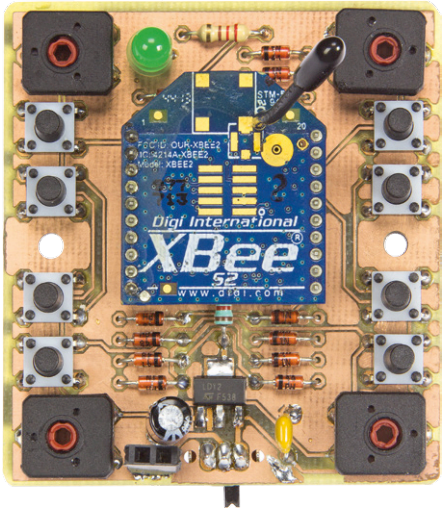
Kondensatoren:
C1,C3 = 1 µF, 6,3 V, Raster 2,5 mm
C2 = 2,2 µ, 6,3 V, Raster 2,5 mm

Halbleiter:
IC1 = MCP1700-3302E/TO
IC2 = 7805
IC3 = ATmega328P-PU, programmiert (150408-43)
LED1 = grün, 3mm

Außerdem:
K1 = 2-polige Stiftleiste, Raster 0,1"
K2, K3 = 1x10-polige Stiftleiste, Raster 2 mm
K4...K7 = 1x3-polige Stiftleiste, Raster 0,1"



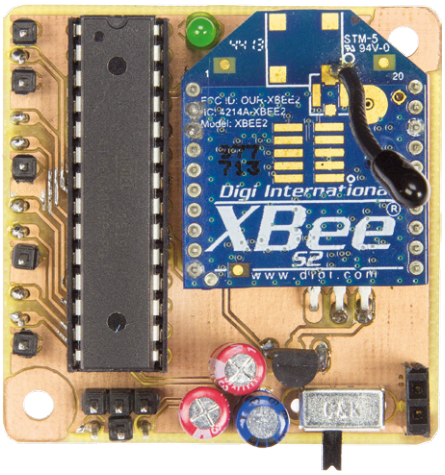
K8 = 2x3-polige Stiftleiste, Raster 0,1"
S9 = Schiebeschalter (OS102011MS2QN1C)
XBee S2C oder XBee-Pro S2C (+ Antenne)
Platine 150408-4



XBee-Moduls hat keine Ahnung, was es mit diesen Daten auf sich hat. Stattdessen gehen die Daten in ein Ohr (Antenne) hinein und aus einem anderen (sprich: dem seriellen Port DOUT) des XBee wieder hinaus. Es gibt zwar XBee-Module, deren Firmware programmierbar ist, so dass auf einen externen Mikrocontroller verzichtet werden könnte, allerdings ist das nur für industrielle Großabnehmer eine wirkliche Option.
Der Mikrocontroller liest die Daten auf seinem seriellen Port und wandelt sie in Signale für die Servos um, die dann an den Stiftleisten K4...K7 zur Verfügung stehen. IC2 wurde übrigens hinzugefügt, um eine saubere 5-V-Versorgung der Servos zu gewährleisten, während die Elektronik mit 3,3 V läuft.
Die Software für den Empfänger wurde mit der Arduino-IDE verfasst und ist daher leicht zu ändern (siehe **Kasten**).

XBee-Sender und XBee-Empfänger verbinden

Bevor die XBee-Module ihre Arbeit aufnehmen können, müssen sie konfiguriert werden. Dazu sind ein Computer und ein USB-Seriell-Adapter (Kabel oder



Platine, siehe Einkaufszettel) erforderlich. Ein XBee-Breakout-Board könnte auch nicht schaden (entweder selbst bauen oder kaufen, siehe Einkaufszettel). Auf dem Computer sollten Sie das kostenlose Programm XCTU [1] installieren, denn nur echte Nerds können AT-Befehle ohne Mithilfe dieses Programms erteilen.
Als Erstes ist das Sendermodul an der Reihe. Verbinden Sie das Modul auf dem Breakout-Board mit dem PC über eine serielle Schnittstelle. Starten Sie XCTU auf dem Computer und klicken Sie auf „Discover radio modules“. Eventuell muss das Modul zurückgesetzt werden, bevor es erkannt wird. Sobald das Modul gefunden wurde, öffnet sich ein Konfigurationsbildschirm. Klicken Sie auf die Schaltfläche „Read“ für die aktuelle Modulkonfiguration (**Bild 3**).

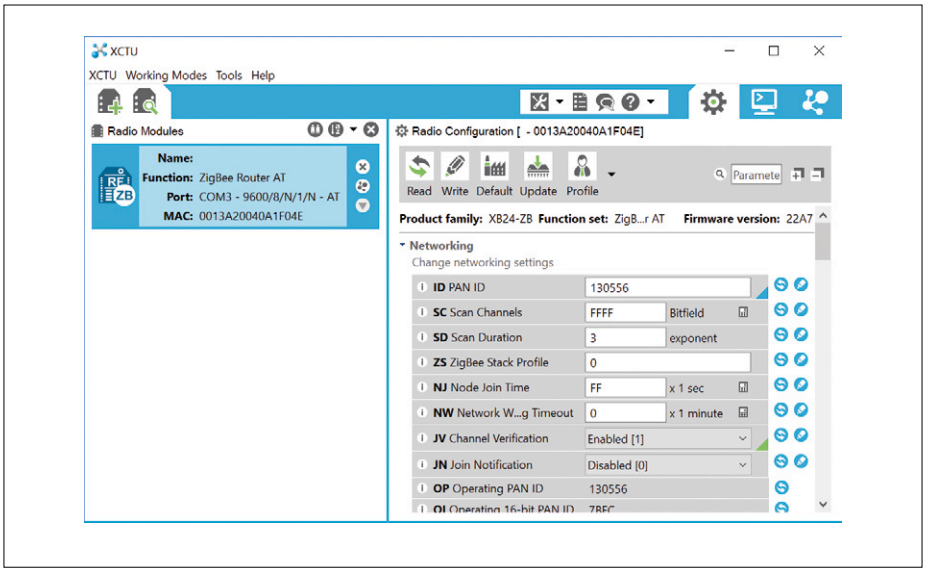


Bild 3. XCTU findet ein XBee-Modul an COM3.

XBee-Sender konfigurieren

Das Modul muss mit dem Funktionssatz „ZigBee Router API“ programmiert werden. Klicken Sie dazu auf „Update“, wählen Sie die Produktfamilie „XB24-ZB“, den Funktionssatz „ZigBee Router API“ und die Firmware-Version „23A7“. Ist dies geschehen, klicken Sie auf „Update“ (**Bild 4**).
Nun kann das XBee-Modul neu konfiguriert werden. Im Bereich „Networking“ muss die „PAN ID“ eingegeben werden, eine Hexadezimalzahl zwischen 0 und 0xFFFF FFFF FFFF FFFF (64 bit). Diese ID verhindert Störungen zwischen verschiedenen Netzwerken. Sie sollten also etwas „Schwierigeres“ eingeben als 1234 und die vergebene ID notieren, da sie auch für das Empfängermodul benötigt wird. Stellen Sie zudem „Channel Verification“ auf „Enabled [1]“.
Scrollen Sie hinunter zum Abschnitt „Addressing“, wo sie die MAC-Adresse des **Empfänger**moduls eingeben müssen. Diese Adresse ist auf das Modul gedruckt, sie kann aber auch mit XCTU gefunden werden. Die ersten (oder höheren) 32 Bits der 64-bit-Adresse gehören in das Feld „Destination Address High“, die unteren 32 Bits in das Feld „Destination Address Low“.
Als Letztes werden, wenn man fast ganz nach unten scrollt, die I/O-Pins des Sendermoduls unter „IO settings“ konfiguriert. **Tabelle 2** und **Tabelle 5** zeigen die von uns benötigten Werte. Setzen Sie unterhalb der „IO settings“ den Wert von „IO Sampling Rate“ auf 64.
Wenn dies geschehen ist, klicken Sie auf „Write“.

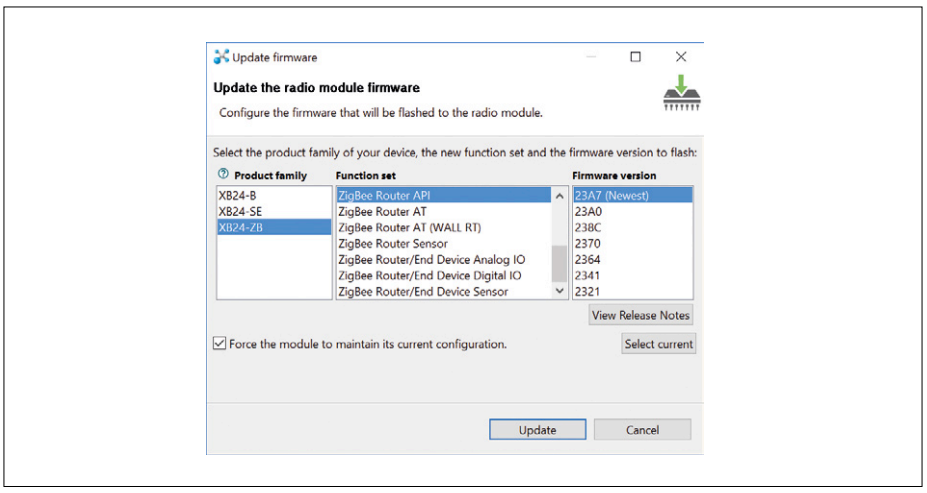


Bild 4. Der XBee-Sender benötigt den „ZigBee router API“-Funktionssatz.

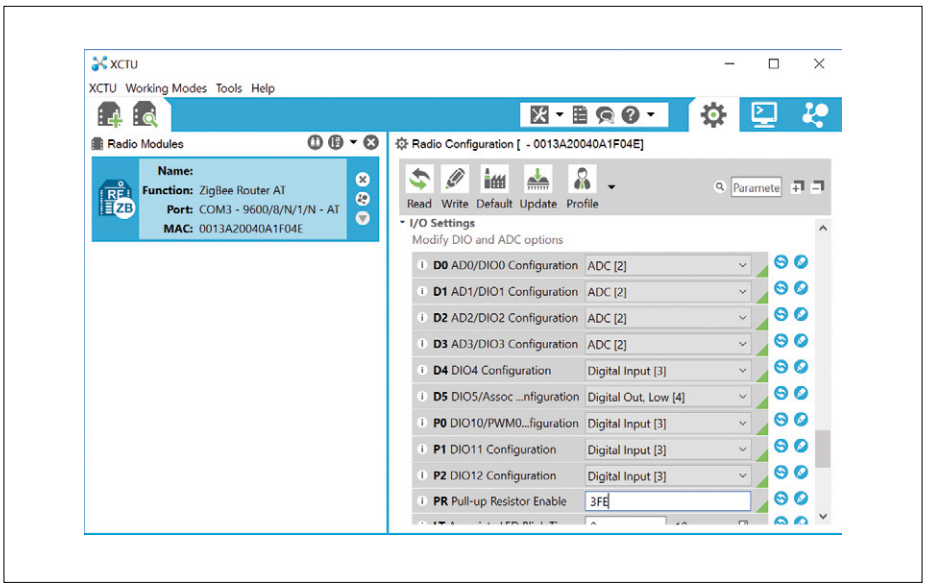
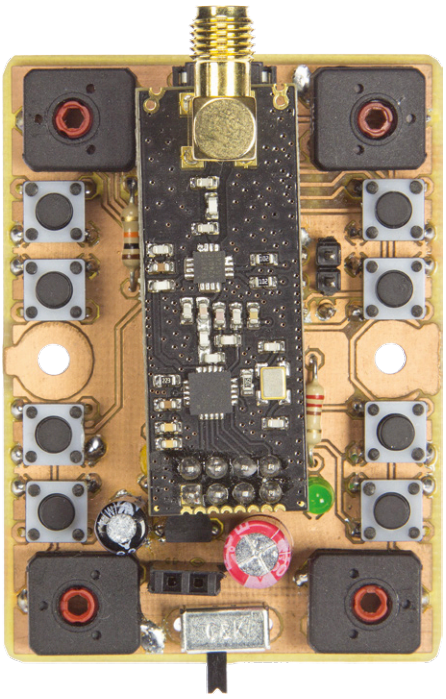


Bild 5. Konfiguration der I/O-Pins des XBee-Senders.



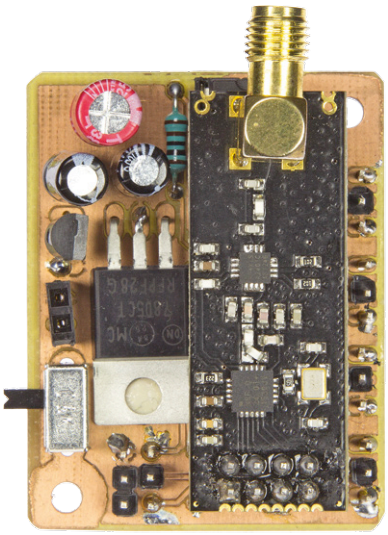
XBee-Empfänger konfigurieren

Das Sendermodul wird entfernt und durch das Empfängermodul ersetzt. Suchen Sie mit XCTU wieder nach dem Modul und drücken Sie auf „Read“, wenn es entdeckt wurde.

Das Empfängermodul wird mit dem Funktionssatz „ZigBee Coordinator API“ geladen. Klicken Sie auf „Update“, wählen Sie als Produktfamilie wieder „XB24-ZB“ und die Firmware-Version „21A7“ aus und eben „ZigBee Coordinator API“. Klicken Sie dann auf „Update“. In der „Networking“-Sektion gibt man die gleiche PAN-ID und ein wenig darunter bei „Addressing“ die Adresse des **Sender**moduls ein. Dann drücken Sie auf „Write“.

Jetzt muss nur noch der Mikrocontroller programmiert werden. Dazu benötigen Sie einen AVR-ISP-Programmierer, der an K8 angeschlossen wird, und natürlich die Firmware, die von [2] heruntergeladen werden kann. Vergessen Sie nicht, die Fuses des Mikrocontrollers wie in **Tabelle 3** einzustellen.

Stecken Sie nun die XBee-Module auf die Platinen (ohne sie zu verwechseln) und schließen Sie die Servos an der Empfängerplatine an. Nun schalten Sie die Stromversorgung beider Platinen ein und warten



Sie ein paar Sekunden ab, bis beide Seiten miteinander verbunden sind. Dann sollten Sie die Servos durch Drehen an den Trimmpotis steuern können. Wenn Sie den Sender an ein serielles Terminal anschließen, werden Rückmeldungen angezeigt.

Und nun mit nRF24-Modulen…

Eine Funkfernsteuerung mit den XBee-Modulen ist sicherlich interessant,

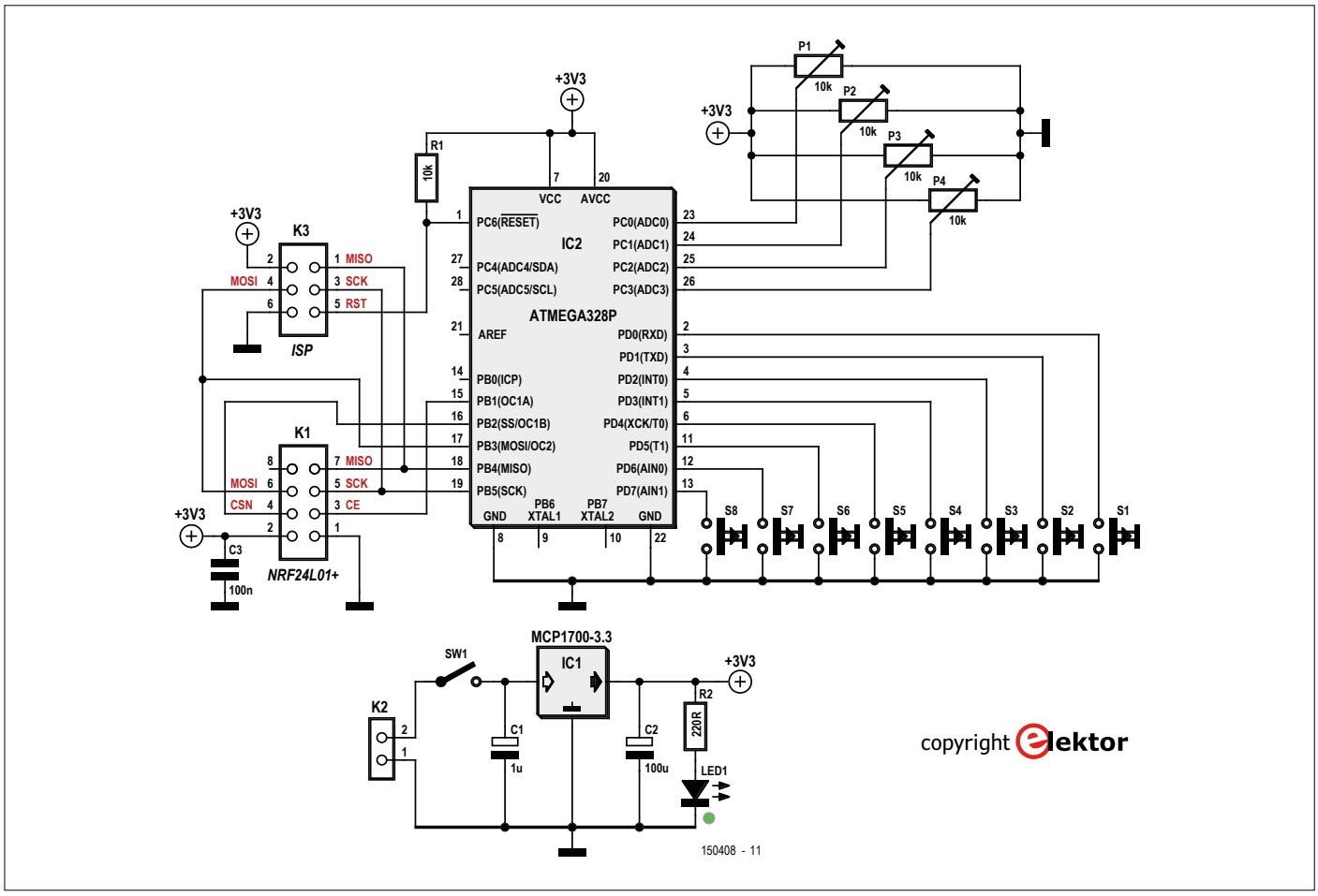


Bild 6. Der nRF24-Sender nutzt einen SPI-Bus zur Kommunikation mit dem Controller.

da die XBee-Hardware eine Vielzahl von Netzwerk-Konfigurationen ermöglicht. Sie ist aber gleichzeitig auch recht teuer. Preisgünstiger sind nRF24L01-Funkmodule, die man im Internet für wenige Euro erwerben kann. Mit einer guten Antenne lässt sich ein Übertragungsabstand ähnlich wie bei einer XBee-Pro-Verbindung erreichen. Obwohl ein zusätzlicher externer Mikrocontroller benötigt wird, sind die Gesamtkosten doch viel geringer als für die XBee-Fernsteuerung.

Tabelle 1. Einstellung der Servo-Offsets.

	'1'	'0'	'TRIM_DOWN'	'TRIM_UP'	Servo Offset
S1	0	0	0	1	Kanal 1 up
S2	0	0	1	0	Kanal 1 down
S3	0	1	0	1	Kanal 2 up
S4	0	1	1	0	Kanal 2 down
S5	1	0	0	1	Kanal 3 up
S6	1	0	1	0	Kanal 3 down
S7	1	1	0	1	Kanal 4 up
S8	1	1	1	0	Kanal 4 down

Der nRF24-Sender

Bild 6 zeigt den Schaltplan des nRF24-basierten Senders (mit den gleichen Trimmpotis und Tastern wie beim XBee-Sender). Das XBee-Modul wurde durch einen ATmega328-Mikrocontroller ersetzt, das nRF24-Funkmodul wird an K1 angeschlossen. Beachten Sie, dass MCU und Funkmodul über einen SPI-Bus und nicht über die asynchrone serielle Schnittstelle miteinander kommunizieren. Da der Controller ausreichend viele I/O-Anschlüsse besitzt, sind die Taster direkt angeschlossen und müssen nicht dekodiert werden, so dass die Dioden entfallen dürfen, genau wie der 1,2-V-Spannungsregler. Alles wird mit 3,3 V betrieben.

Der nRF24-Empfänger

Die Schaltung des nRF24-Empfängers in **Bild 7** und der fertig aufgebaute Prototyp in **Bild 8** unterscheiden sich nur marginal von der XBee-Version. Anstelle des XBee-Moduls wurde lediglich der Verbinder K1 für den Anschluss des nRF24-Moduls eingesetzt. Auch hier kommunizieren Funkmodul und Controller über SPI statt über die asynchrone serielle Schnittstelle.

nRF24-Sender und nRF24-Empfänger verbinden

Dies ist noch einfacher als bei den XBee-Modulen, da lediglich die Firmware für Sender und Empfänger in den entsprechenden Controllern untergebracht werden muss. Im **Kasten** finden Sie dazu detaillierte Anweisungen. Sollte es zu Störungen mit anderen nRF24-Fernsteuerungen kommen, so muss die Netzwerkadresse des Systems geändert werden. In beiden Sketches findet sich oben eine Zeile wie diese:

```
uint64_t address[] = {0xFFFFFFFF,
   0xFFFFFFFE};
```

Ersetzen Sie in einer der beiden Dateien die obigen Werte durch zwei andere, aber unterschiedliche 64-Bit-Werte (was immer Sie möchten) und kopieren Sie die geänderte Zeile in die andere Datei. Neu kompilieren, neu programmieren, fertig!

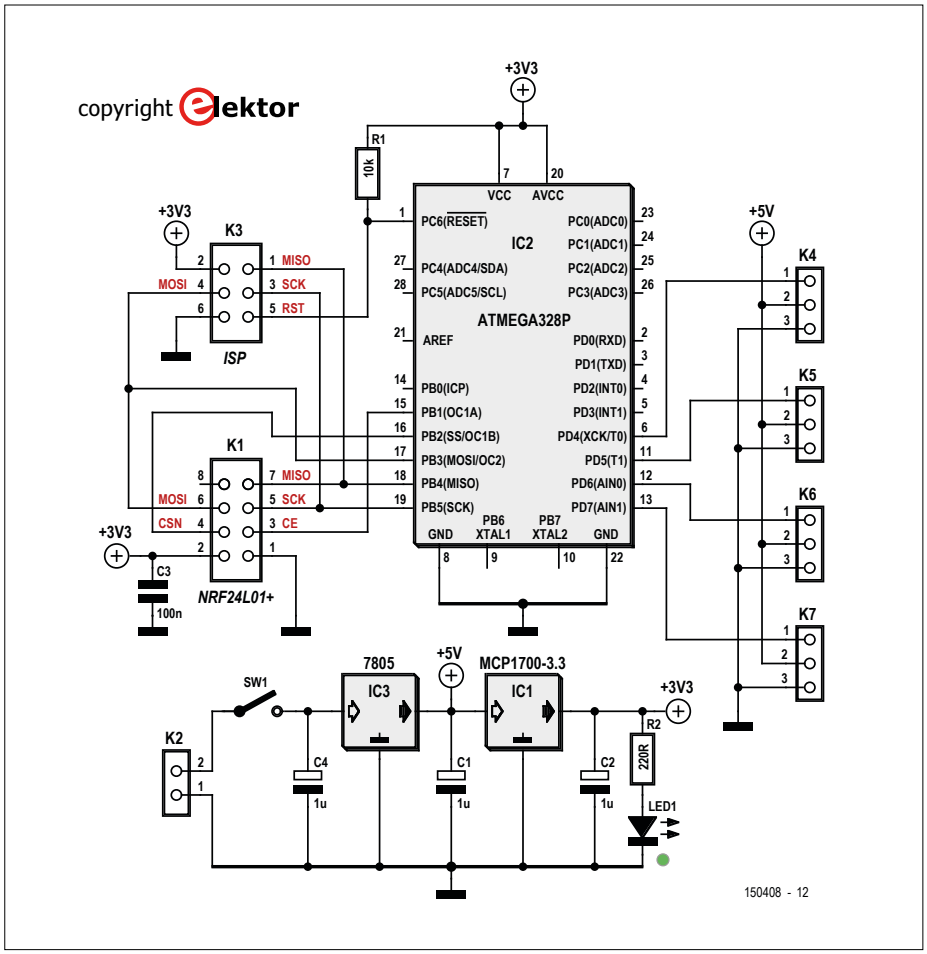


Bild 7. Der nRF24-Empfänger ist beinahe identisch mit dem XBee-Empfänger.

Tabelle 2. So werden die I/O-Ports des XBee-Senders eingestellt.

I/O	Value
D0	ADC [2]
D1	ADC [2]
D2	ADC [2]
D3	ADC [2]
D4	Digital Input [3]
D5	Digital Out, Low [4]
P0	Digital Input [3]
P1	Digital Input [3]
P2	Digital Input [3]
PR	3FE

Tabelle 3. Fuse-Einstellungen für die AVR-Mikrocontroller.

Fuse	Wert
Low	0xE2
High	0xDA
Extended	0x05

Schließen Sie die Servos an der Empfängerplatine an, schalten Sie die Stromversorgung beider Platinen ein und warten Sie ein paar Sekunden ab, bis beide Seiten miteinander verbunden sind. Dann sollten Sie die Servos durch Drehen an den Trimmpotis steuern können. Wenn Sie den Sender an ein serielles Terminal anschließen, werden Rückmeldungen angezeigt.

Zusammengefasst...

Es wurden zwei verschiedene Fernsteuerungssysteme mit einer möglichen Zwei-Wege-Datenkommunikation vorgestellt, XBee und nRF24. Welche am besten für Sie geeignet ist, hängt von Ihrer Anwendung und Ihrem Budget ab. Die Software stellt nur die Basis-Funktionen einer Servosteuerung zur Verfügung, die erweitert und für aufwändigere Systeme modifiziert werden muss. Zum Glück ist dies leicht, da die Programme open source und für die Arduino-IDE geeignet sind. ◀

(150408)

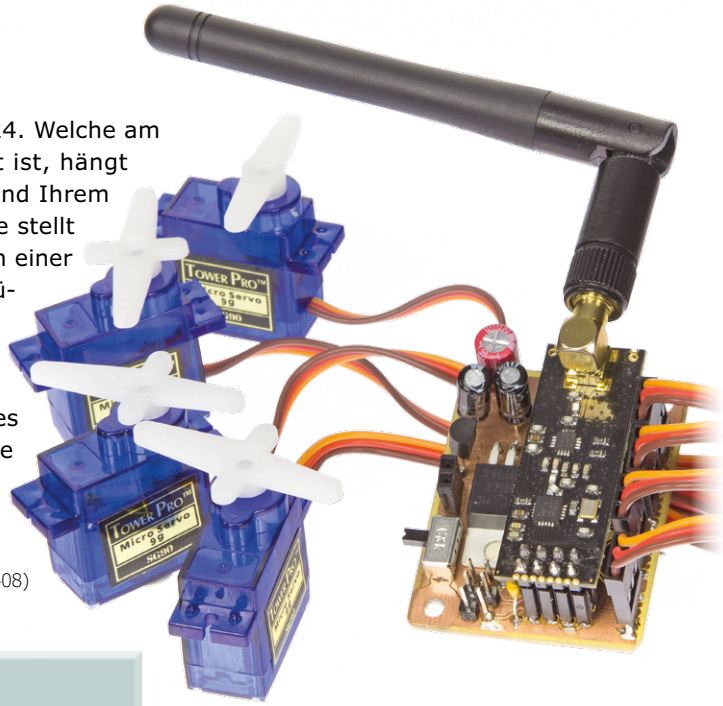


Bild 8. Unser Prototyp des nRF24-Empfängers steuert vier angeschlossene kleine Servos.

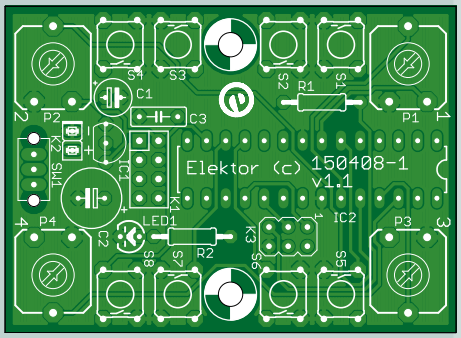
STÜCKLISTE 150408-1 nRF24-SENDER

Widerstände:
P1...P4 = 10 k Trimpoti
R1 = 10 k
R2 = 220 Ω

Kondensatoren:
C1 = 1 µ, 6,3 V, 2,5 mm
C2 = 100 µ, 6,3 V, 3,5 mm
C3 = 100 n

Halbleiter:
IC1 = MCP1700-3302E/TO
IC2 = ATmega328P-PU, programmiert (150408-41)
LED1 = grün, 3 mm

Außerdem:
K1 = 2x4-polige Stiftleiste, Raster 0,1"
K2 = 2-polige Stiftleiste, Raster 0,1"



K3 = 2x3-polige Stiftleiste, Raster 0,1"
S1...S8 = Taster, 6x6 mm
SW1 = Schiebeschalter (OS102011MS2QN1C)
nRF24L01+PA+LNA (+ Antenne)
Platine 150408-1

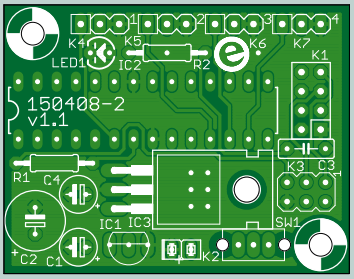
STÜCKLISTE 150408-2 nRF24-EMPFÄNGER

Widerstände:
R1 = 10 k
R2 = 220 Ω

Kondensatoren:
C1,C4 = 1 µ, 6,3 V, 2,5 mm
C2 = 100 µ, 6,3 V, 3,5 mm
C3 = 100 n

Halbleiter:
IC1 = MCP1700-3302E/TO
IC2 = ATmega328P-PU, programmiert (150408-42)
IC3 = 7805
LED1 = grün, 3 mm

Außerdem:
K1 = 2x4-polige Stiftleiste, Raster 0,1"



K2 = 2-polige Stiftleiste, Raster 0,1"
K3 = 2x3-polige Stiftleiste, Raster 0,1"
K4...K7 = 1x3-polige Stiftleiste, Raster 0,1"
SW1 = Schiebeschalter (OS102011MS2QN1C)
nRF24L01+PA+LNA (+ Antenne)
Platine 150408-2

Weblinks

[1] www.digi.com/products/xbee-rf-solutions/xctu-software/xctu

[2] www.elektormagazine.de/150408

EINKAUFSZETTEL

nRF24-Version

→ **150408-1**
Sender-Platine

→ **150408-2**
Empfänger-Platine

→ **150408-41**
Sender-Mikrocontroller programmiert

→ **150408-42**
Empfänger-Mikrocontroller programmiert

XBee-Version

→ **150408-3**
Sender-Platine

→ **150408-4**
Empfänger-Platine

→ **150408-43**
Empfänger-Mikrocontroller programmiert

Tools

→ **140374-91**
T-Board Wireless

→ **150387-1**
Hi-speed-USB-UART

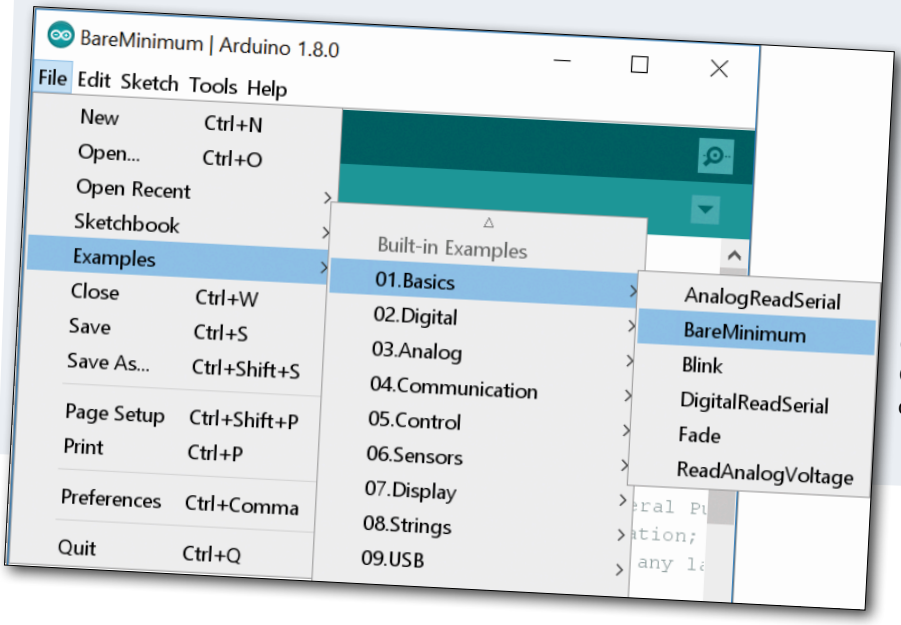
HomeLab Helikopter

Zusammengestellt von **Clemens Valens** (Elektor-Labor)

Arduino-IDE-Hack: Neue Sketch-Vorlage

Wenn Sie die Arduino-IDE öffnen oder einen neuen Sketch über File➞New erstellen, entdecken Sie im Editor bereits leere Skelette für die Funktionen Setup und Loop. Wenn Sie wie ich sind und die Art und Weise, wie diese Skelette geschrieben sind, gar nicht mögen, so müssen Sie auch jedes Mal die Skelette bearbeiten, was fast genauso zeitaufwendig ist wie das Schreiben einer neuen Hülle von Grund auf. Wie lässt sich diese Vorlage ändern? Öffnen Sie dazu die Datei `<arduino>\examples\01.Basics\BareMinimum\BareMinimum.ino` in einem externen Texteditor, ändern Sie sie nach Belieben und speichern Sie sie. Das nächste Mal, wenn Sie einen neuen Sketch erstellen wollen, finden Sie dann Ihr eigenes Sketchgerüst vor. Sie können dies nicht in der IDE selbst ändern, weil die IDE versuchen würde, das Template vor dem Überschreiben zu schützen. Eine benutzerdefinierte Vorlage ist auch eine großartige Möglichkeit, automatisch Codezeilen (wie `Serial.begin (115200);`) oder auch Lizenztext vorzugeben.

(160354-a)

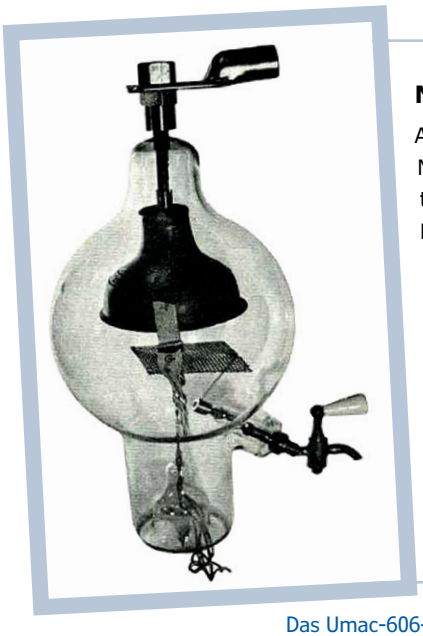


Ein Muss für jedes Heimlabor

Alle Maker lieben Gadgets und nützliche Helferlein. Deshalb halten wir immer ein Auge offen für gute Ideen, um den ohnehin schon beträchtlich hohen Haufen auf ihren Schreibtischen anwachsen zu lassen. Dieses Mal möchten wir Ihnen ein solarbetriebenes (öko!) Werkzeug vorstellen, das der Maker, so er einmal seine Werkbank verlässt (ja, soll vorkommen!), stets mitführen muss! Vielleicht wandelt er in der Wildnis, wo er von Mücken und Schnaken angegriffen wird und dadurch den rechten Weg und schließlich völlig die Orientierung verliert. Wenn dies geschieht, wird der Kompass ihm helfen, den Rücksturz zum Labortisch sicher zu finden, ohne dass er dabei von lästigen Insekten gepiesackt wird. Die praktische Karabinerform passt zu jeder Gürtelschlaufe, so dass das Tool leicht zu tragen ist, wohin man auch geht.

(160354-d)

Die solar-betriebene Mückenscheuche Digisonic 12D8 mit Kompass, erhältlich in jedem guten Webshop.



Das Umac-606-Phantasatron.

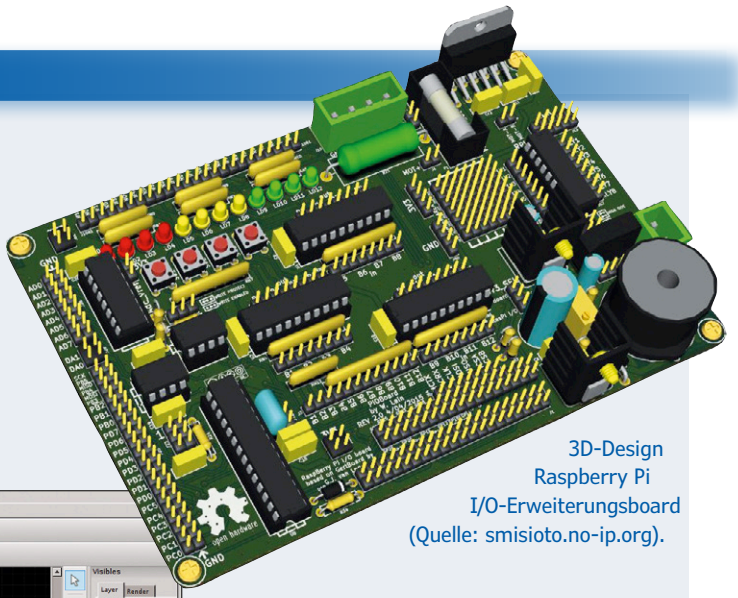
Noch mehr Bemerkenswerte Bauteile

An anderer Stelle in dieser Ausgabe kann man etwas über das Phantastron lesen, einen speziellen Multivibrator mit sehr präzisem Timing, der im Zweiten Weltkrieg erfunden wurde. Das Phantastron ist eine sehr clevere Schaltung, aber ich bin sicher, dass, hätte ihr Schöpfer Alan Dower Blumlein das Phantastron Umac 606 gekannt, seine Idee noch viel viel cleverer gewesen wäre. Das spiralförmig gestrahlte Phantasatron Umac 606, das etwa zur gleichen Zeit auf den Markt kam, besaß eine Infernal-Anode mit uriniertem Wolframwendel, der durch ein spezielles (geheimes) Verfahren mit dreifach destilliertem Einfach-Isotop-Uran unter vollständiger Entfernung der Neutronen erzeugt wurde. Das Gitter war aus der seltenen Erde Senilium gebaut. Dieses Material ist für seine völlige Emissionsfreiheit bekannt. Ein doppelt gepumptes Vakuum gewährte eine klare Sicht auf die nicht emittierende, dreifach veredelte tauchspulartige Platte. Wegen seiner einzigartigen selbstspülenden Konstruktion blieb diese Röhre während seiner gesamten Nutzungsdauer nutzbar. Weitere Einzelheiten entnehmen Sie aber bitte jetzt dem Datenblatt.
www.tubecollectors.org/archives/606.pdf

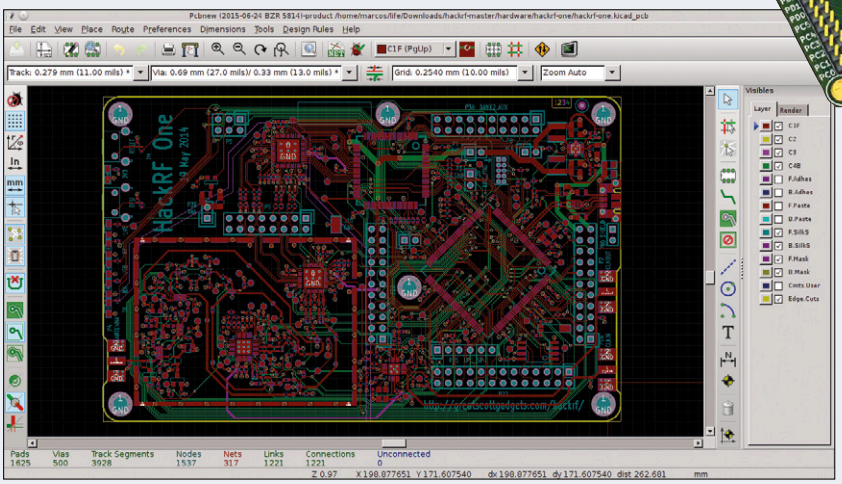
(160354-c)

3D-Bibliothek für KiCad-EDA

In den vergangenen Jahrzehnten hat die Open-Source-Software-Community für sehr viele, sehr gute Produkte gesorgt, von denen einige auch für kommerzielle Anwender völlig kostenlos sind. Geschaffen und unterhalten werden die Programme von Tausenden von Enthusiasten in Teams und als Einzelpersonen, die sich nur selten im wirklichen Leben kennenlernen. Man kann nur die Energie und die Mühe bewundern, die diese Leute bereit sind, für die Projekte aufzuwenden, ohne dafür etwas zu erwarten. Linux ist wohl das Erste, was einem in den Sinn kommt, aber es gibt noch viele andere



3D-Design Raspberry Pi I/O-Erweiterungsboard (Quelle: smisioto.no-ip.org).



KiCad PCB-Editor PcbNew (Quelle: kicad-pcb.org).

Programme wie zum Beispiel KiCad-EDA, eine plattformübergreifende Automatisierungs-Suite zur Schaltungsentwicklung. Laut Wikipedia wurde das Projekt 1992 von Jean-Pierre Charras entworfen. Meine ersten Erfahrungen mit KiCad stammen etwa aus dem Jahr 2003. Damals konnte man es „benutzen“, aber es war für meinen Job noch nicht geeignet. KiCad hat sich aber weiterentwickelt und ist heute, nach 25 Jahren, ein professionelles Schaltungs- und Platinenlayoutprogramm mit 3D-Betrachtungsmöglichkeit.

Zu solch erfolgreichen Gemeinschaftsprojekten tragen viele Menschen bei. Im Falle von KiCad betrifft dies vor allem Bauteilbibliotheken und Tutorials, aber auch funktionale Erweiterungen wie den interaktiven „Push-and-Shove“-Router, der vom Hardware& Timing-Team des CERN entwickelt wurde. Meine eigenen Beiträge zu KiCad sind eher marginal – nur dieser kurze Text, mit dem ich der Welt diese frohe Kunde unterbreite. Kürzlich entdeckte ich eine italienische Website, die von einem Walter aus Padua gepflegt wurde und eine sehr große 3D-Bauteilbibliothek für KiCad anbietet. Daneben erläutert die Website auch einige Wege, um schöne 3D-Bauteilmodelle zu entwerfen. Wenn Sie also jemals ein eigenes Modell entwerfen wollen, schauen Sie einmal in Padua vorbei!

http://smisioto.no-ip.org/elettronica/kicad/kicad-en.htm
http://kicad-pcb.org/

(160354-b)

Kann man „Persönlichen Assistenten“ trauen?

Alles fing vor einigen Jahren mit Siri von Apple an. Dieser „intelligente“ persönliche Assistent war plötzlich im iOS von Apples Smartphones und Tablets eingebaut. Natürlich konnte das die Konkurrenz nicht auf sich sitzen lassen und zog nach. Microsoft brachte Cortana, Google seinen Assistant und Samsung dann Viv. Alle haben ihre Stärken und Schwächen, doch werden sie wohl in Zukunft immer besser. Überraschenderweise aber hat Alexa, der Nachzügler von Amazon, das Potential, sie alle zu schlagen. Zuerst als Bestandteil des extrem erfolgreichen sprachgesteuerten Lautsprechers „Echo“ samt Verwandtschaft (Echo Dot und Tap) zeigte sich Alexa auf der CES 2017 in Las Vegas schon in gut 30 neuen Produkten. Die Sprachbegabung erschöpft sich dabei nicht darin, Text hörbar zu machen, sondern in der Steuerbarkeit durch gesprochene Sprache. Man sagt „spiele Elvis“ und die Kiste spielt Elvis.

Der Erfolg von Alexa hat wohl mit der Offenheit des Konzepts zu tun. Wo Siri und Cortana zumindest zu Anfang komplett abgeschottet waren, gibt es für Alexa von Anfang an eine offene Plattform, was diverse Entwickler dazu motiviert hat, „skills“ hinzuzufügen. Ein Skill ist einfach eine neue Funktion, etwa die Bestellung von Pizza oder die Änderung einer Beleuchtungsfarbe, indem man dies einfach befiehlt. Google Assistant ist ebenfalls offen und erlaubt die Entwicklung von „actions“ durch Dritte. Durch den Erfolg von Alexa sahen sich Microsoft und Apple vor kurzem genötigt, einen Entwicklerzugang zu Cortana und Siri zu schaffen. Anders als Siri und Cortana, die zumindest teilweise auf die Rechenleistung der Mobilgeräte setzen, vertraut Alexa voll auf die Server von Amazon. Damit lässt sich auch einfachere Elektronik mit Sprachsteuerung ausstatten. Natürlich muss für die Verbindung zur Cloud eine permanente Internetverbindung vorhanden sein. Ein System, das Alexa-Services nutzt, muss immer zuhören und erwacht dann, wenn das Trigger-Wort „Alexa“ gesprochen wird. Ab dann beginnt eine Audio-Aufzeichnung und parallel die Übertragung über das Internet zum analysierenden und interpretierenden Server. Die Daten werden außerdem für weitere Zwecke (etwa die Verbesserung der Spracherkennung) gespeichert. An diesem Punkt kommt die Privatsphäre und ihre mögliche Verletzung ins Spiel. Ein Gerät, dass gesprochene Sprache aus privaten Unterhaltungen ins Internet überträgt, ist prinzipiell dazu in der Lage, die Privatsphäre zu verletzen, was nicht nur bei staatlichen Stellen, sondern auch bei Hackern die Begehrlichkeit wecken könnte, eine solch interessante Datenbasis zu infiltrieren und unangenehme Dinge daraus zutage fördern. Laut Amazon wurden bisher alle Anfragen bezüglich eines Zugriffs auf diese Daten zurückgewiesen. Doch wie lange noch? Und was passiert insgeheim?

ROGERS © 2015 Pittsburgh Post-Gazette. Nachdruck mit Erlaubnis von ANDREWS MCMEEL SYNDICATION. Alle Rechte vorbehalten.

Zwar könnten Anwender ihre Aufzeichnungen löschen, doch wer würde das tun, wenn man gleichzeitig damit die Leistungsfähigkeit seines intelligenten Assistenten beträchtlich herabsetzt? Sind Sie bereit, ihre Privatsphäre der Bequemlichkeit zu opfern, eine Pizza bestellen zu können, ohne dafür den Finger in die Wählscheibe eines Telefons stecken zu müssen?

(160354-e)

2016 wurden 25.000.000.000 Mikrocontroller verkauft.

Onyx Connect ist die erste Firma, die Smartphones und Tablets in Afrika herstellt.

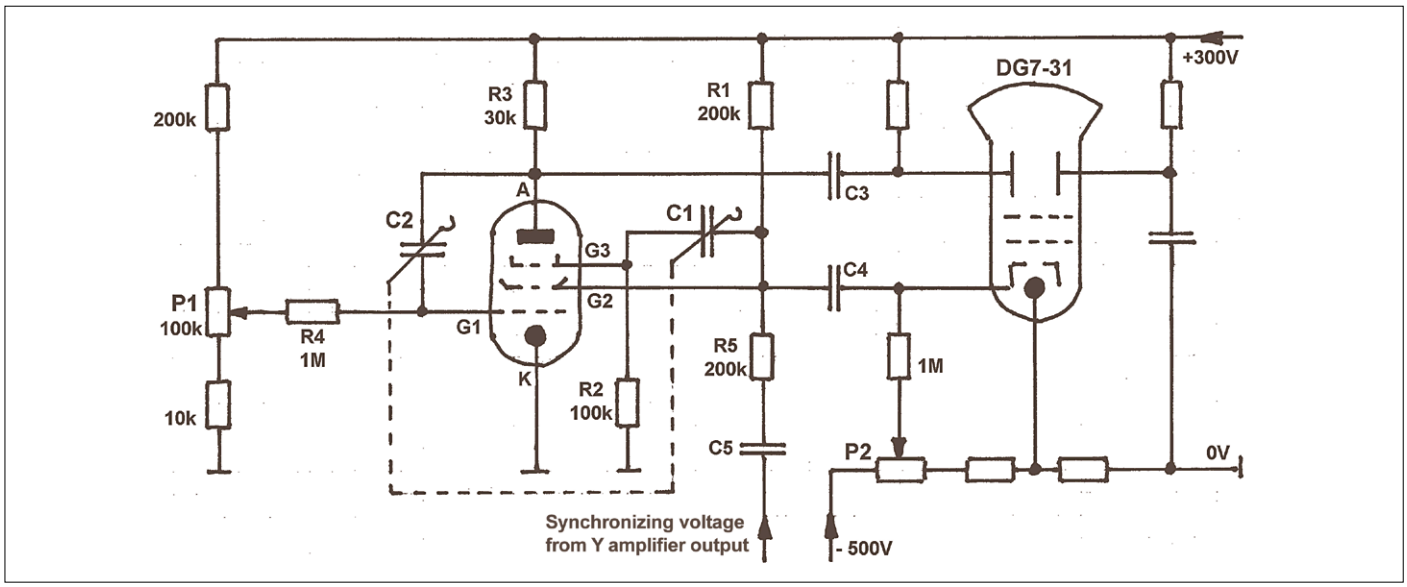
Möchten Sie mitmachen? Bitte senden Sie Ihre Kommentare, Anregungen, Tipps und Tricks an labs@elektor.com.

Das Phantastron

Bemerkenswerte Bauteile

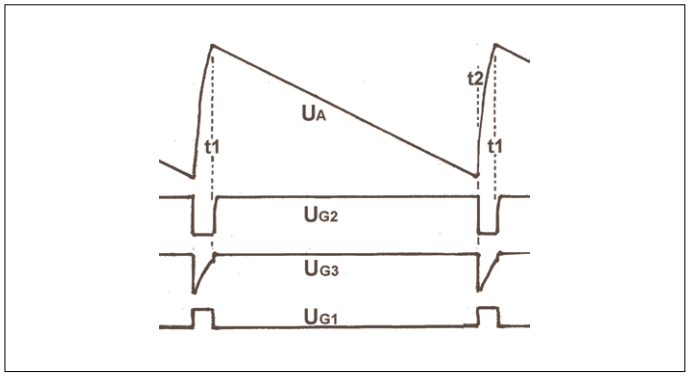
Von **Lutz Bergmann**

Wie konnte es passieren, dass eine solch extrem lineare und genaue Schaltung in Vergessenheit gerät? Sie verdient hohe Anerkennung!



Die Namen einiger elektronischer Bauelemente enden auf -tron, wie zum Beispiel Ionitron, Magnetron, Thyratron und Klystron. Jedes kann als bemerkenswert bezeichnet werden, weshalb das Klystron bereits in dieser Reihe behandelt wurde, genau wie das Dekatron und das Trochotron... Im Gegensatz zu den anderen -trons ist der Hauptdarsteller dieses Artikels nicht ein Bauteil, sondern eine sehr eigenartige und spezielle Art von Schaltung. Sie wurde 1940 von einer bemerkenswerten Persönlichkeit entworfen: Alan Blumlein, der Erfinder der Stereophonie und vieler weiterer Errungenschaften in den Bereichen Telekommunikation, Tonaufzeichnung, Fernsehen und Radar [1].

Im Phantastron wird eine Pentode in einer Weise angesteuert, die man in herkömmlichen Verstärkerschaltungen sorgfältig vermeidet. Ein merkwürdiger Effekt wird zur Erzeugung von Sägezahnspannungen eingesetzt [2][3]: Wenn die Plattenspannung (Anodenspannung) kleiner wird als die Schirmgitterspannung, steigt der Strom im Schirmgitter einer Pentode stark an. Die Elektronen werden statt zur Anode hin mehr auf das Schirmgitter gezogen, da dies die höhere Spannung aufweist. Wird das Schirmgitter über einen Widerstand mit Strom versorgt, so beginnt dann die bisher von der Anodenspannung unabhängige Schirmgitterspannung zu fallen. Diese negative Änderung kann über den Kondensator C1 auf das Bremsgitter übertragen werden, das dann als zusätzliches Steuergitter fungiert und den Fluss des letzten verbleibenden Anodenstroms durch die nun negative Steuerspannung hemmt. Der Anodenstrom wird so abrupt abgeschaltet. Die Spannung an der Anode kann noch einmal schnell ansteigen. Mit dieser bistabilen Anordnung kann ein so genannter Miller-Integrator zu einem freilaufenden Sägezahn-Oszillator erweitert werden.



Beginnen wir mit dem Ende einer Ausphase, genannt t1. Die Anodenspannung ist maximal, da der Anodenstrom Null ist. Da der Koppelkondensator C1 die Bremsgitterspannung nur für eine begrenzte Zeit negativ halten kann, beginnt der Anodenstrom wieder zu fließen. Der Spannungsabfall am Widerstand R3 steigt an und die Anodenspannung U_A fällt. Diese Spannungsänderung bewirkt einen zirkulierenden Strom an C2, der wiederum zu einer negativen Spannungsänderung am Steuergitter G1 führt, was den Anstieg des Anodenstroms bremst. Zwischen den Strömen durch C2 und R4 entsteht ein Gleichgewicht. Da der Strom durch R4 konstant ist, gilt das gleiche für C2. Die Anodensspannung fällt zeitlich wie gewünscht linear ab.

Dieser Effekt wurde 1919 von John Milton Miller [4] entdeckt. Das Maß, mit dem die Anodenspannung absinkt, kann mit dem Potentiometer P1 im Verhältnis 1:10 stufenlos und in Schritten durch Auswahl des Kondensators C2 eingestellt werden. Dieser Vorgang wiederholt sich, so er nicht gestört wird, bis die Anoden- kleiner als die Schirmgitterspannung und der Anodenstrom dadurch gehemmt wird. Die Anodenspannung steigt noch einmal an, wenn auch nicht mit beliebiger Geschwindigkeit, da C2 wieder aufgeladen werden muss. Hierbei spielt die Zeitkonstante C2 x R3 eine Rolle. Die Spannung auf dem Bremsgitter G3 muss dazu ausreichend lang negativ bleiben. C1 muss deshalb auf C2 abgestimmt sein. Wird C2 durch einen Schalter verändert wird, muss das gleiche für C1 geschehen.

Nach 1945 wurden solche Sägezahnoszillatoren mit nur einer einzigen Pentode bei einfachen Oszilloskopen eingesetzt, da damals Vakuumröhren noch sehr teuer waren. So wurde es möglich, alle für die horizontale Auslenkung einer Kathodenstrahlröhre (CRT) notwendigen Signale bereitzustellen. Die Anordnung konnte ohne zusätzliche Bauteile 200 V_{SS} erzeugen. Über den Kondensator C3 konnte man diese Wechselspannung an eine der beiden horizontalen Ablenkplatten legen (asymmetrische Ansteuerung der DG7-31 mit 7 cm Schirmdurchmesser). Für den Wehnelt-Zylinder [5] der Kathodenstrahlröhre konnte ein negativer Spannungsimpuls über Kondensator C4 aus dem Schirmgitter abgegriffen werden, um eine Austastlücke zu ermöglichen.
Ein solch einfaches Oszilloskop war nicht besonders komfortabel. Die horizontale Auslenkung war grundsätzlich freilaufend, so dass die Sägezahn-Oszillatorfrequenz auf die Frequenz des zu untersuchenden Signals eingestellt werden musste, um ein statisches Bild zu erhalten. Zur Synchronisation wurde der Ausgang des vertikalen y-Verstärkers über das RC-Glied R5/C5 mit dem Schirmgitter verbunden.
Mit dem Aufkommen der triggerbaren horizontalen Auslenkung und den fallenden Preisen der Vakuumröhren verschwand das Phantastron schon vor langer Zeit von der Bildfläche. Gleichzeitig haben Transistoren und später die integrierten Schaltkreise Vakuumröhren verdrängt, während Sweep-Generatoren eine viel einfachere Triggerung ermöglichten.
Weitere Informationen finden Sie im Internet, einschließlich einer Halbleiterversion [6] und der Verwirrung über das große Geheimnis dieser Schaltung [7].

(160281)

Weblinks

[1] Alan Blumlein: https://de.wikipedia.org/wiki/Alan_Blumlein

[2] Miller-Transistron: www.r-type.org/articles/art-135.htm

[3] Miller-Zeitbasis: www.r-type.org/articles/art-136.htm

[4] Millereffekt: https://de.wikipedia.org/wiki/Millereffekt

[5] Wehneltzylinder: https://de.wikipedia.org/wiki/Wehneltzylinder

[6] Radartutorial: www.radartutorial.eu/17.bauteile/bt52.de.html

[7] „Is Security Dead?“ in Wireless World (S. 380, S. 56 des PDF): http://americanradiohistory.com/Archive-Wireless-World/40s/Wireless-World-1946-11.pdf

Anzeige

NEU

pico® Technology

THE PICOSCOPE 4444

SEE THE DIFFERENCE

A NEW STANDARD IN DIFFERENTIAL MEASUREMENT

- 20 MHz bandwidth, 12 - 14 bit resolution
- 4 fully differential inputs
- 1000 V CAT III probes
- Low voltage probes and current clamps
- 256 MS deep memory
- Measures non-grounded (floating) signals
- Rejects common-mode noise

Für weitere Informationen besuchen Sie www.picotech.com/PS505

Email: Sales@picotech.com .Irrtümer und Druckfehler vorbehalten.
Für aktuelle Preise kontaktieren Sie bitte vor der Bestellung Pico Technology.

IoT-Gateway und drahtlose Endknoten

Teil 2: Die Software

Im zweiten Teil dieses Beitrags stellt der Autor die Software des von ihm entwickelten Domotik-Systems vor. Zur Erinnerung: Die „Endknoten" (Sensoren und Aktoren) kommunizieren drahtlos im 868-MHz-Band mit einem zentralen Gateway. Das Gateway arbeitet mit dem Protokoll MQTT, um die Messdaten einem OpenHAB-Server zu übergeben. Der Server verarbeitet und visualisiert die Messdaten. Zur Hardware, die ausführlich im ersten Teil dieses Beitrags beschrieben wurde, gehört natürlich auch Software, die das System zum Leben erweckt.

Von **Hennie Spaninks** (NL)

OpenHAB (*Open Home Automation Bus*) ist eine Softwarelösung für die Automatisierung in der Gebäudetechnik [1], sie wurde bereits im ersten Teil dieses Beitrags [2] erwähnt. Dazu kommen Arduino-Sketche, die der Autor selbst geschrieben hat. Dabei konnte er auf diverse Open-Source-Bibliotheken zurückgreifen. Software läuft auf folgenden Plattformen:

End Node
Arduino-Sketch für die Endknoten: Programmcode zum Bedienen der Funkverbindung, Lesen der Sensoren und Steuern der Ausgänge.

Gateway
Arduino-Sketch für das Gateway: Programmcode zum Bedienen der Funkverbindung und zum Auswerten von Nachrichten. Ein MQTT-Client tauscht die Nachrichten mit dem Broker aus.

Domotik-Server (Raspberry Pi)
Auf dem Raspberry Pi laufen sowohl der MQTT-Broker (Mosquitto) als auch der Domotik-Controller OpenHAB.

Zum Kompilieren des Codes für den Arduino kommt die Arduino-Entwicklungsumgebung zum Einsatz, erweitert durch Bibliotheken wie nachfolgend angegeben. Beim Raspi sollte man auf Modell 3 zurückgreifen, einige Features laufen auch auf dem Raspberry Pi 2. Das verwendete Betriebssystem ist Raspbian.

Endknoten

Der Programmcode für die Endknoten ist unter [3] verfügbar. Bei allen Endknoten ist der Kern identisch. An dieser Stelle soll der Endknoten DHT [4] betrachtet werden. Der Endknoten besitzt einen digitalen Eingang (Drucktaster) an Pin 8, einen digitalen Ausgang (Relais oder LED) an Pin 9 sowie einen Luftfeuchte- und Temperatursensor an Pin 4. Nach Drücken des Tasters kann der Endknoten lokal zwei Funktionen ausführen: Starten eines Timers und Setzen des Ausgangs auf High. Reset des Ausgangs nach Ablauf des Timers. Das Timer-Intervall kann mit Device 7 eingestellt werden (siehe Tabelle 2 im ersten Teil des Beitrags). Der Wert 0 blockiert den Timer.

Umschalten des Ausgangs nach Aktivieren des Drucktasters. Damit kann der Ausgang lokal von Hand geschaltet werden. Voraussetzung ist, dass Device 6 auf ON steht.

Unabhängig von den Funktionen wird nach Niveau-Änderungen am digitalen Eingang (Pin 8) eine Statusmeldung über Device 40 ausgegeben. Die Messdaten des Luftfeuchte- und Temperatursensors können über Device 48/49 gelesen werden. Auch periodische Messdaten-Übertragungen sind möglich.
Der Programmcode bindet drei Bibliotheken ein:
Bibliothek für das Modul RFM69 von Felix Ruso [5],
SPI-Bibliothek zum Steuern des SPI-Bus,
DHT-Bibliothek zum Lesen des Sensors DHT11.
Der Programmcode beginnt mit den Parametern für die Konfiguration (ab Zeile 60):

```
#define NODEID 2
   // unique node ID within the closed network
#define GATEWAYID 1
   // node ID of the Gateway is always 1
#define NETWORKID 100
   // network ID of the network
#define ENCRYPTKEY "xxxxxxxxxxxxxxxx"
   // 16-char encryption key; same as on Gateway!
   //#define DEBUG
   // uncomment for debugging
#define VERSION "DHT V2.2"
   // this value can be queried as device 3
```

Die `NODEID` weist dem Endknoten eine eindeutige Adresse im Funknetzwerk zu. Die ID wird auch im MQTT-Programmcode verwendet, um den Endknoten zu adressieren. Das Gateway hat immer die `NODEID` 1. Ein Endknoten unterhält eine Funkverbindung ausschließlich mit dem Gateway. Zwischen den Endknoten ist keine Funkverbindung möglich.
Die `NETWORKID` ermöglicht das Definieren geschlossener Netzwerke, so dass nur Knoten innerhalb des gleichen Netzwerks miteinander kommunizieren können. Dieses Projekt arbeitet mit einem einzigen geschlossenen Netzwerk, die Netzwerk-ID muss folglich beim Gateway und allen Endknoten identisch sein.
Der `ENCRYPTKEY` ist der Schlüssel, mit dem die Funkverbindungen verschlüsselt werden. Der Schlüssel besteht aus 16 Zeichen, auch er muss bei allen Knoten des Netzwerks identisch sein.
Der Parameter `DEBUG` ist ein Schalter, der den Debugging-Modus aktiviert. Wenn dieser Parameter nicht definiert ist, weil die Programmzeile mit „//“ auskommentiert wurde, wird keine Debugging-Information erzeugt. Nach Löschen der Zeichen „//“ ist der Modus aktiv, dann werden Statusmeldungen über den seriellen Port ausgegeben.
Zum korrekten Steuern des Funkmoduls RFM69 sind noch einige Parameter nötig:

```
#define FREQUENCY RF69_868MHZ
#define IS_RFM69HW   // uncomment only for RFM69HW!
#define ACK_TIME 50  // max # of ms to wait for an
   ack
```

Der Parameter `FREQUENCY` gibt die benutzte Frequenz an, in Europa 433 MHz oder 868 MHz, und `IS_RFM69HW` kennzeichnet die High-Power-Version des Funkmoduls. `ACK_TIME` ist die Zeit, die nach Senden einer Nachricht bis zum Eintreffen einer Bestätigung gewartet wird.
Ab Zeile 78 folgen Definitionen für den Sensor DHT11, für die

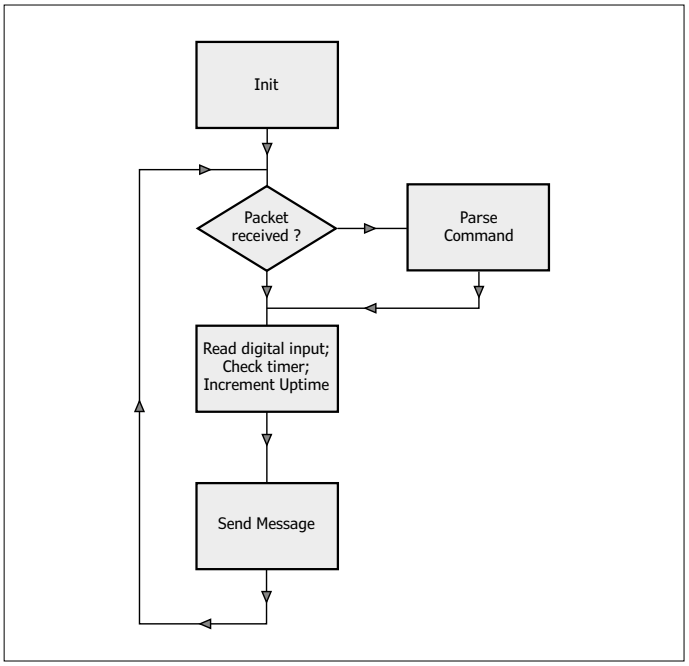


Bild 1. Flussdiagramm für einen Endknoten

belegten Pins und die verwendeten Variablen. Die Kommentare im Quellcode weisen darauf hin, welche Funktionen die einzelnen Variablen haben. Bei Zeile 135 schließt sich die Programm-Initialisierung an (`setup()`), dort wird der Ausgang konfiguriert und das Funkmodul wird initialisiert. Das zugehörige Flussdiagramm ist in **Bild 1** dargestellt.
Zuerst prüft der Programmcode (Zeile 167), ob das Funkmodul RFM69 ein Datenpaket empfangen hat. Wenn dies zutrifft, wird die Funktion `parseCmd` aufgerufen. Abhängig von der Device-ID und vom R/W-Kommando im empfangenen Datenblock wird das zutreffende Sende-Flag gesetzt. Dann wird geprüft, ob der Drucktaster aktiviert wurde und der Timer abgelaufen ist. Anschließend wird der Uptime-Counter inkrementiert. Sobald das Intervall für die periodische Datenübertragung beendet ist, werden die Sende-Flags dieser Parameter gesetzt.
Schließlich wird in Zeile 235 die Funktion `sendMsg` aufgerufen. Abhängig vom aktiven Sende-Flag stellt `sendMsg` die Daten für die Funkübertragung bereit, setzt das Sende-Flag zurück und sendet das Datenpaket durch Aufrufen der Funktion `txRadio()`. Falls die Temperatur oder die Luftfeuchte die zu übertragenden Größen sind (Device 48 und 49), wird in den Zeilen 478 und 485 der Sensor DHT11 gelesen.

Gateway

Das Gateway hat folgende Funktionen:

- Empfange eine MQTT-Nachricht, werte sie aus, generiere und sende die Funknachricht.
- Empfange eine Funknachricht, werte sie aus, generiere eine MQTT-Nachricht und sende sie an alle.
- Empfange und beantworte Anfragen, die das Gateway betreffen, beispielsweise Uptime, Versionsnummer und Ähnliches.

Das Gateway ist mit dem Netzwerk über eine stationäre Ethernet-Leitung verbunden, es erhält seine Netzwerkinformationen

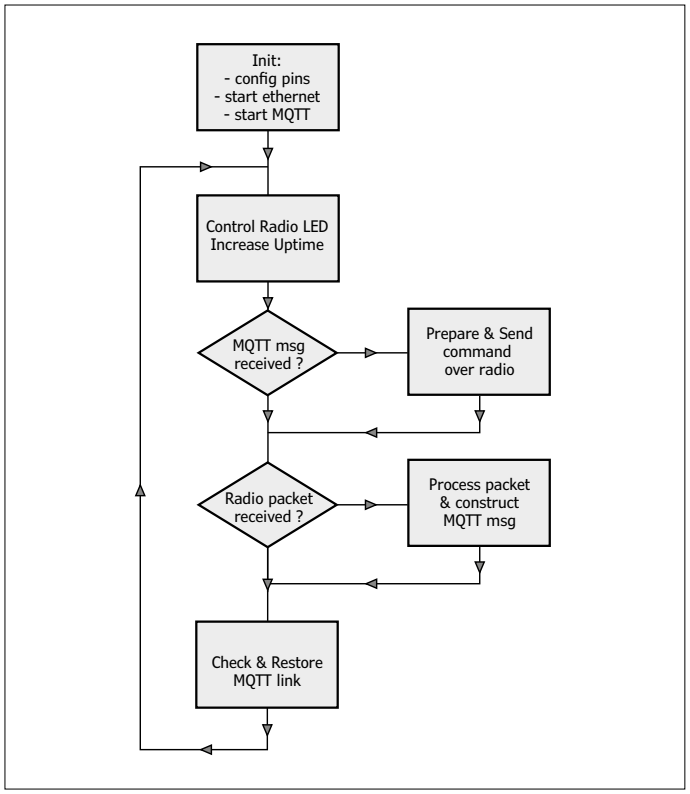


Bild 2. Flussdiagramm für das Gateway.

über DHCP. Am Gateway befinden sich zwei LEDs: Die Funk-LED leuchtet auf, wenn die Funkverbindung aktiv ist. Die MQTT-LED leuchtet auf, wenn eine Verbindung mit Mosquitto besteht.
Da der Programmcode des Gateways vergleichsweise viel Speicher belegt, ist die Debug-Funktion zweigeteilt: Ein Teil dient zum Debuggen der Funkverbindung, der andere zum Debuggen der Verbindung mit dem MQTT-Broker. Es kann immer nur ein Teil aktiv sein, anders lässt sich der Programmcode nicht kompilieren.
Die Programmbeschreibung bezieht sich auf die aktuelle Version 2.4, sie steht unter [6] im Netz.
Der Gateway-Sketch bindet folgende Bibliotheken ein:

- *RFM69*
- *SPI*
- *Ethernet*
- *PubSubClient* [7]

Zum Kompilieren des Gateway-Sketches mussten/müssen einige Standardeinstellungen geändert werden:
Beim Gateway liegen zwei SPI-Devices am SPI-Bus, bei beiden ist die Adresse 10 voreingestellt, was zu Konflikten führen würde. Das RFM-Modul erhält daher die SPI-Device-Adresse 8, während das Ethernet-Modul die Adresse 10 behält. Diese Konfiguration steht in Zeile 80 des Gateway-Programmcodes.
Das RFM-Modul signalisiert den Empfang von Funknachrichten durch Interrupts. Solange der Ethernet-Treiber (w5100.h) auf dem SPI-Bus aktiv ist, werden die Interrupts nicht adäquat abgearbeitet. Die Folge ist, dass das Gateway hängen bleibt. Deshalb wird von der Lösung Gebrauch gemacht, die Martin Harazinov vorgeschlagen hat (siehe [8]). Die Datei w5100.h, die unter *../Documents/Arduino/libraries/Ethernet/src/utility* zu finden ist, wird modifiziert. Ab Zeile 342 werden folgende Zeilen durch Statements mit `cli()`und `sei()`ergänzt:

```
#else
inline static void initSS()   { DDRB |= _BV(2); };
inline static void setSS()    { cli(); PORTB &= ~_
   BV(2); };
inline static void resetSS()  { PORTB |= _BV(2);
   sei(); };
#endif
```

Dadurch werden die Interrupts während der Ethernet-Aktivitäten auf dem SPI-Bus blockiert.

Der Programmcode des Gateway-Sketchs beginnt mit einem Abschnitt von Deklarationen, ähnlich wie beim Endknoten. In dem Abschnitt werden sämtliche Parameter für die Funkverbindungen eingestellt, außerdem werden folgende Parameter für die Ethernet-Verbindung festgelegt:

- `Ip`, für die IP-Adresse des Gateways. Dies ist ein Fall-back, das Gateway verwendet DHCP, ihm wird dynamisch eine IP-Adresse zugewiesen.
- `Mac`, für die MAC-Adresse des Ethernet-Anschlusses. Innerhalb eines IP-Subnetzes muss die Adresse eindeutig sein. Das ist wichtig, sobald sich mehrere Arduinos im Netzwerk befinden.
- `Mqtt_server`, die Adresse des Mosquitto-Servers. Davon wird nachfolgend noch die Rede sein.

Das Flussdiagramm des Gateway-Programmcodes ist in **Bild 2** dargestellt. Die Initialisierung beginnt mit Zeile 135. Die Ausgänge werden für die LEDs konfiguriert, die Ethernet-Verbindung wird gestartet, und anschließend wird die Verbindung mit dem MQTT-Broker aufgebaut.
In Zeile 129 initialisiert das Programm den MQTT-Client. Dabei wird ein Abonnement angefordert, das in der Funktion `mqtt_subs` (Zeile 382) definiert ist. Topic des Abonnements ist *home/rfm_gw/sb/#* (Zeile 121), so dass alle SouthBound-Nachrichten empfangen werden. Sobald MQTT-Nachrichten eintreffen, wird die Variable `mqttToSend` auf TRUE gesetzt. Das Hauptprogramm prüft zuerst, ob die Funk-LED abgeschaltet werden muss, dann wird der Uptime des Gateways inkrementiert.
Das Programm prüft in Zeile 223 mit der Variablen `mqttToSend`, ob Nachrichten über das Funknetzwerk gesendet werden müssen. Dies geschieht in Zeile 248 durch die Funktion `sendMsg`. Falls nötig, wird die Übertragung bis zu fünf Mal im Abstand einer halben Sekunde wiederholt (so lange bis der Endknoten den Empfang bestätigt).
Dann wird in Zeile 225 geprüft, ob eine Funknachricht von einem Endknoten empfangen wurde. Falls dies zutrifft, wird in Zeile 288 die Funktion `processPacket` aufgerufen. Eine MQTT-Nachricht des Gateways bezieht sich immer auf die NorthBound-Richtung (also hin zum OpenHAB-Server). Die Nachricht besteht aus der `nodeID` und der `deviceID`, sie wird in Zeile 303 dem String `buff_topic` übergeben. Abhängig von der `deviceID` wird entschieden, welche Daten der Funknachricht gültig sind (Integer, Float, Status oder String) und welche Struktur für die MQTT-Nachricht erforderlich ist. Zum Schluss enthält der String `buff_mess` die MQTT-Nachricht.

Zum Schluss wird in Zeile 227 geprüft, ob die MQTT-Verbindung noch aktiv ist. Sollte dies nicht der Fall sein, wird in Zeile 230 in Abständen von zwei Sekunden versucht, die Verbindung neu zu starten. Wenn die Verbindung hergestellt ist, wird in Zeile 235 die MQTT-LED eingeschaltet.

OpenHAB

In **Bild 3** ist die Architektur von OpenHAB mit dem charakteristischen Event-Bus dargestellt. Über diesen Bus werden Events zwischen den Eingangs- und Ausgangsschnittstellen und den übrigen OpenHAB-Komponenten ausgetauscht. Durch die Struktur ist es möglich, mehrere IO-Schnittstellen (*bindings*) parallel zueinander zu betreiben. Zum Beispiel sind spezifische Bindings für unterschiedliche Protokolle realisierbar. Gegenwärtig existieren Bindings unter anderem für Smarte Messsysteme, Sonos, Hue-Lampen von Philips, für die Nest-Thermostaten sowie für Automobile von Tesla. Die Website von OpenHAB [9] enthält eine Übersicht.

Installation

Zum Einsatz kommt OpenHAB Version 2 auf einem Raspberry Pi 3 mit einer SD-Speicherkarte, Kapazität 16 GB. Die Installation lässt sich bequem mit OpenHABian bewerkstelligen. Das ist ein Image mit einem Linux-Minimalsystem, kombiniert mit diversen Installationsskripts. Das Image kann von OpenHAB.org [10] heruntergeladen werden. Mit einem passenden Tool, beispielsweise mit dem WinDiskImager [11], wird das Image auf die SD-Speicherkarte übertragen. Wenn die SD-Karte im Raspberry Pi steckt, startet nach dem nächsten Booten die Installation. Dann ist etwas Geduld nötig, die Installation dauert einige Zeit.
Bei der Installation von OpenHAB wird auch *Samba* installiert. Mit Samba ist der Zugriff auf die Konfigurationsdateien von OpenHAB über das Netzwerk möglich, so dass Konfigurationsänderungen vorgenommen werden können.

Um Mosquitto zu installieren, wird nach Abschluss der Installation das Konfigurationstool gestartet:

```
pi@openHABianPi: sudo openhabian-config
```

Wenn das Menü in **Bild 4** erscheint, wird mit Auswahl 22 Mosquitto installiert. Wichtig ist hier, dass dem Raspberry Pi stets die gleiche IP-Adresse zugewiesen wird. Das lässt sich durch Modifizieren der Datei */etc/network/interfaces* erreichen. Entsprechend der gewünschten Konfiguration werden `address` und `gateway` angepasst, zum Beispiel:

```
auto eth0
    iface eth0 inet static
        address 192.168.2.7
        netmask 255.255.255.0
        gateway 192.168.2.254
```

Nach einem Reboot sind die neuen Einstellungen wirksam.

Konfiguration

Damit OpenHAB läuft, muss die MQTT-Anbindung korrekt konfiguriert sein. Auf dem Raspberry Pi heißt die anzupassende Datei */etc/openhab2/services/mqtt.cfg*. Die Broker-URL ist der Parameter, der angepasst werden muss, er ist wie folgt zu ändern:

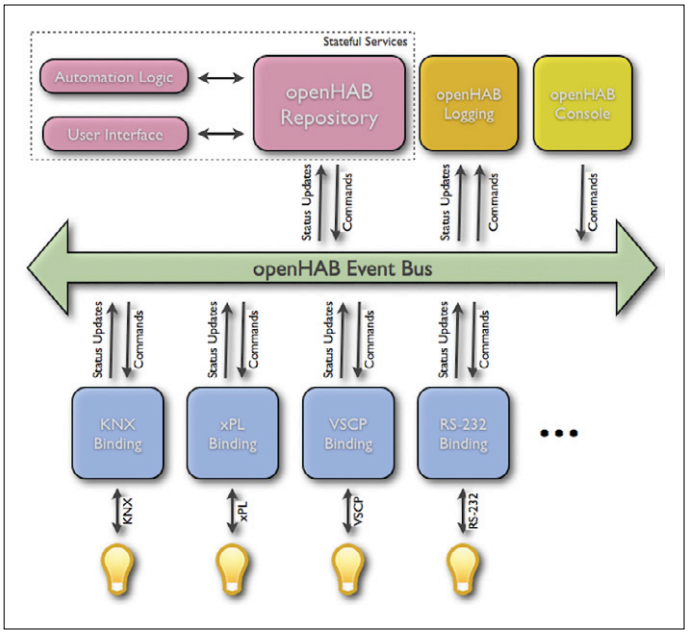


Bild 3. OpenHAB-Architektur (Quelle: OpenHAB.org).

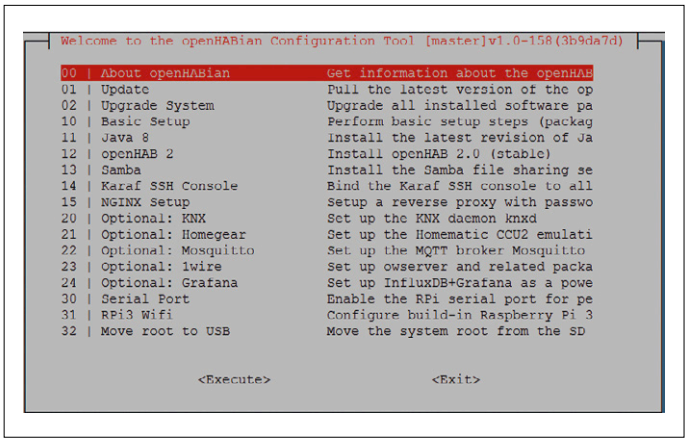
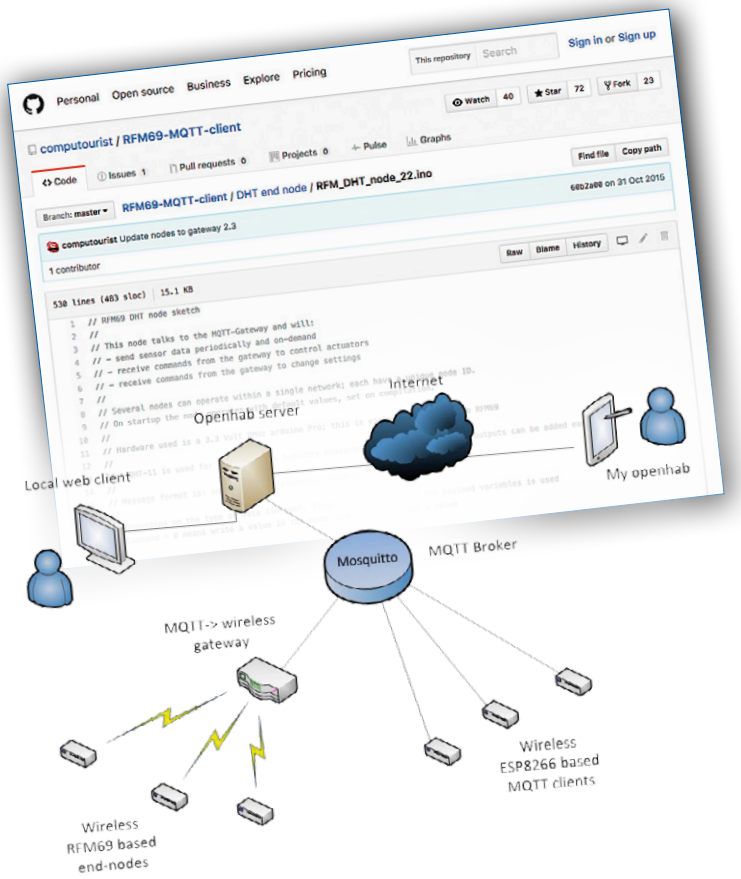


Bild 4. OpenHABian-Konfiguration.

```
# Define your MQTT broker connections here for use
   in the MQTT Binding or MQTT
# Persistence bundles. Replace <broker> with an ID
   you choose.
# URL to the MQTT broker, e.g. tcp://localhost:1883
   or ssl://localhost:8883
mosquitto.url=tcp://localhost:1883
```

Dies ist ein Verweis für den Mosquitto-Server auf den Localhost. Zu beachten ist hier, dass *mosquitto* in dieser Konfiguration der Name (ID) des Brokers ist. In der *Item*-Konfiguration (siehe folgender Text) wird auf diese ID verwiesen, um die Kommunikation herzustellen. Port 1883 ist der Default-Port für den MQTT-Informationsaustausch.
OpenHAB verwendet drei Konfigurationsdateien, um ein System zu beschreiben:

- *Items* im Ordner */etc/openhab2/items*, darin stehen die Definitionen der verwendeten Parameter und ihre Kopp-



lung mit der IO.

- *Sitemap* im Ordner */etc/openhab2/sitemaps*, beschreibt das Layout der Benutzerschnittstelle.
- *Rules* im Ordner */etc/openhab2/rules*, gibt an, welche Regeln ausgeführt werden müssen (Scripting).

Als Beispiel soll die Konfiguration eines Endknotens vom Typ DHT mit *nodeID* 2 dienen, der so konfiguriert ist, dass die Temperatur, die Luftfeuchte und der Ausgangsstatus periodisch übertragen werden. Im Item-Ordner (*/etc/openhab2/items*) wird eine Datei *home.items* mit dem Inhalt aus **Listing 1** erstellt. Das OpenHAB-Wiki [12] erklärt, wie die Konfiguration eines Items aufgebaut sein muss. Anschließend wird im Sitemap-Ordner */etc/openhab2/sitemaps* die Datei *home.sitemap* mit folgendem Inhalt erstellt:

```
sitemap home label="MyHome"

Frame label="Controls" {
       Switch item=OUT2 label="Myhome lamp"
   icon="light" }

Frame label="Sensoren" {
       Text item=TEMP2
       Text item=HUM2
       Text item=RSSI2   }
```

In der ersten Zeile steht der Name dieser Konfiguration (home), so dass beim Zugriff auf OpenHAB mit einem Browser oder einer App die korrekte Sitemap geladen werden kann. Dann folgt eine Definition, um den Ausgang von Knoten 2 zu visualisieren, sowie ein Frame, das die Temperatur, die Luftfeuchte und die Feldstärke wiedergibt. Schließlich muss noch eine Datei *home.rules* mit den folgenden Zeilen erstellt werden:

```
//
//  refresh rules
//

rule "refresh RSSI2"   // periodically refresh
   value for signal strength
    when
       Time cron "0 0/1 * * * ?"
   then
       sendCommand(getRSSI2, "READ")
end
```

Dies bewirkt, dass einmal in der Minute die MQTT-Nachricht *READ* zum Device 2 auf Knoten 2 gesendet wird, um die Feldstärke zu lesen.
Nun sind erste Tests an der Reihe. In das Adressfeld eines Browsers wird eingetragen:

```
http://192.168.xx.xx:8080/basicui/app?sitemap=home
```

Natürlich muss als IP-Adresse die zutreffende Adresse eingetragen werden. Im Ergebnis erscheint das Fenster, das **Bild 5** zeigt. Möglicherweise sind noch nicht alle Daten verfügbar (kenntlich am Eintrag „-“), doch nach dem ersten Refresh des Knotens müssen auch diese Daten erscheinen.

Debuggen

Was ist zu tun, wenn das System nicht wie erwartet arbeitet?

Läuft MQTT?

Wenn die MQTT-Verbindung zwischen dem Gateway und dem Broker hergestellt ist, leuchtet die MQTT-LED auf. Dies bedeutet, dass Nachrichten vom Gateway zum Mosquitto-Server übertragen werden. Der folgende Schritt ist das Installieren eines MQTT-Testclients (zum Beispiel MQTT.fx [13] oder MQTTLens [14] aus dem Chrome-Webstore). MQTT.fx bietet einen Tab *broker status* an, über den Informationen zum Mosquitto-Ser-

Listing 1. OpenHAB-Item-Konfiguration eines Endknotens für Temperatur und Luftfeuchte.

```
Number RSSI2 "RSSI [%d dBm]" {mqtt="<[mosquitto:home/rfm_gw/nb/node02/dev02:state:default]"}
String getRSSI2 "get RSSI" {mqtt=">[mosquitto:home/rfm_gw/sb/node02/dev02:command:*:default]"}
Number TEMP2    "Temperatuur [%.1f °C]" {mqtt="<[mosquitto:home/rfm_gw/nb/node02/dev48:state:default]"}
Number HUM2    "Vochtigheid [%.1f %%]"{mqtt="<[mosquitto:home/rfm_gw/nb/node02/dev49:state:default]"}
Switch OUT2 "zolder node" <PushBtn> {mqtt=">[mosquitto:home/rfm_gw/sb/node02/dev16:command:ON:ON],>[mos
   quitto:home/rfm_gw/sb/node02/dev16:command:OFF:OFF]"}
```

ver abgefragt werden können. Nach Abonnieren von *home/rfm_gw/#* müssen Nachrichten erscheinen, bei erfolgreichem Verlauf sind darunter auch Nachrichten des Endknotens. Falls das nicht zutrifft, werden das Gateway oder der Endknoten mit der DEBUG-Option kompiliert. Dann wird ein PC mit dem seriellen Port verbunden und PUTTY [15] gestartet. Dort müssen noch der benutzte COM-Port und die Geschwindigkeit 115200 Baud eingestellt werden. Die Ausgabe des Knotens oder des Gateways gibt Aufschluss darüber, ob eine Verbindung besteht und welche Daten übertragen werden. Wenn über den MQTT-Client Kommandos zum Knoten gesendet werden (beispielsweise *READ* nach */home/rfm_gw/node02/dev03*), dann lässt sich daraus schlussfolgern, wo die Datenübertragung unterbrochen ist.

Läuft OpenHAB?

Im Fall, dass die MQTT-Verbindung korrekt arbeitet, ist das Überprüfen von OpenHAB der nächste Schritt. Die Log-Dateien können wichtige Informationen enthalten. Das Kommando

```
> tail -f /var/log/openhab2/events.log
```

hat zur Folge, dass die letzten zehn Zeilen des Event-Logs angezeigt werden. Sichtbar müssen eintreffende Updates des Knotens oder der Benutzerschnittstelle sein. Sinnvoll ist auch ein Blick in */var/log/openhab2/openhab.log*, denn dort werden OpenHAB-System-Events protokolliert.

Rundblick

Eine naheliegende Erweiterung ist der Zugang über das Internet, zum Beispiel mit *myopenhab*. Zu OpenHAB gehört eine App, die über ein sicheres Gateway und das Internet mit dem heimischen OpenHAB-System kommuniziert (siehe [1]).

Bild 5. Browser-Fenster.

OpenHAB kann auch Grafiken generieren, die Änderungen der Messdaten abhängig von der Zeit darstellen. Dazu muss eine lokale Datenbank aktiviert werden, außerdem ist festzulegen, welche Messdaten erfasst werden sollen. Das kann entweder lokal oder über den Zugriff auf externe Datenbanken geschehen. Weiterführende Informationen stehen auf der Website von OpenHAB.org [16].

Der beschriebene Endknoten misst Temperatur und Luftfeuchte, er kann einen digitalen Ausgang schalten. Durch Anpassen der Hard- und Software des Endknotens eröffnen sich viele weitere Möglichkeiten. Auf Github [3] werden Lösungen mit einem LC-Display und ein RFID-Leser vorgestellt [17].

Parallel zu der hier vorgestellten Lösung wurde ein Endknoten auf Basis des WLAN-Moduls ESP8266 entwickelt. Dieser Knoten stellt über WLAN eine direkte Verbindung mit dem MQTT-Broker und OpenHAB her. Der Entwurf wird auf Github [18] beschrieben.

Eine Fundgrube für weitere Anwendungen ist das Homeautomation-Forum [19], dort werden PIR-Sensoren, Rollladen-Steuerungen, Niveau-Sensoren und vieles mehr präsentiert. ⏮

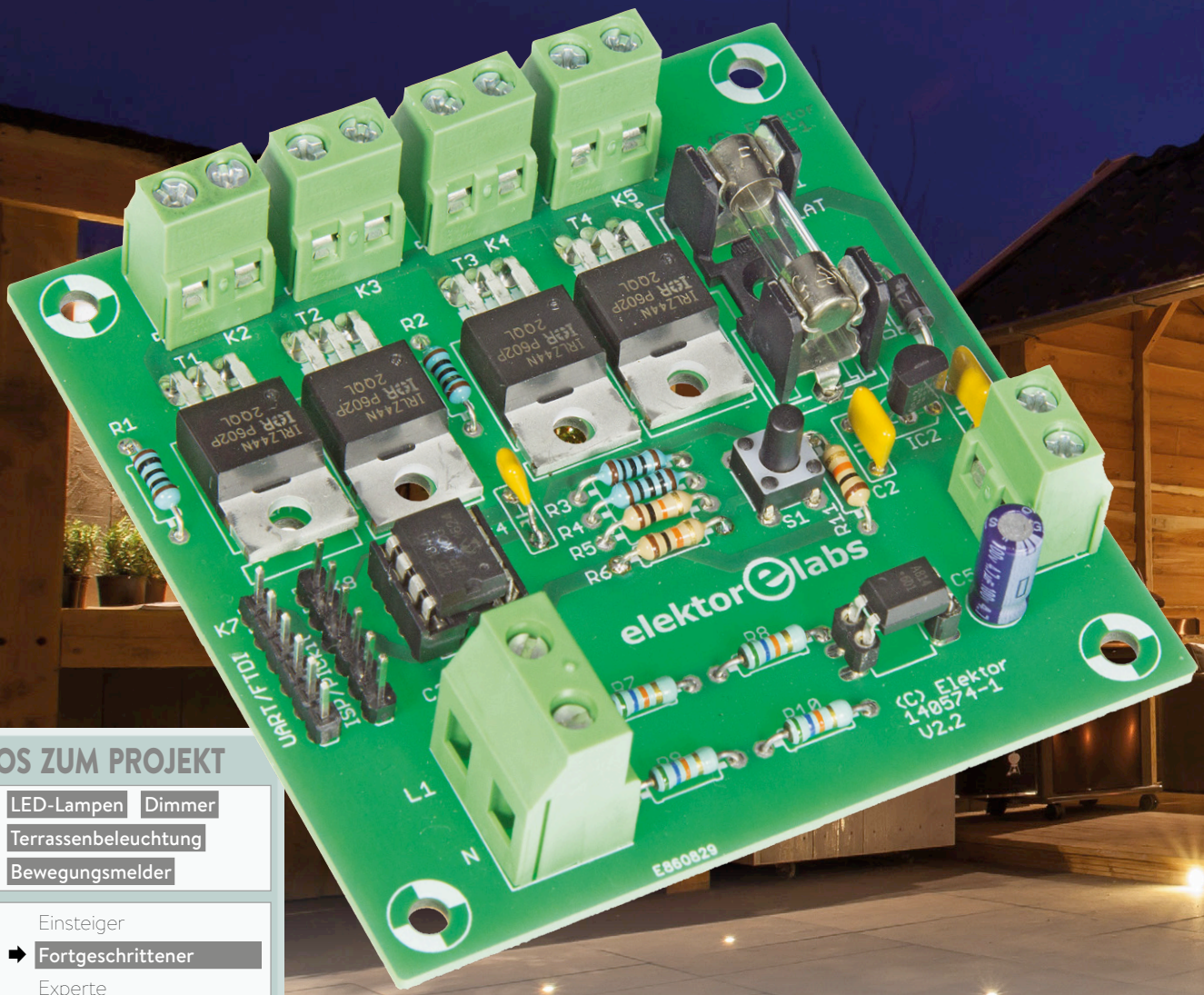
(160318)gd

Weblinks

[1] www.myopenhab.org
[2] www.elektormagazine.de/150085
[3] https://github.com/computourist/RFM69-MQTT-client
[4] https://github.com/computourist/RFM69-MQTT-client/blob/master/DHT%20end%20node/RFM_DHT_node_22.ino
[5] https://github.com/LowPowerLab
[6] https://github.com/computourist/RFM69-MQTT-client/blob/master/Gateway_2.4/RFM_MQTT_GW_24.ino
[8] http://harizanov.com/2012/04/rfm12b-and-arduino-ethernet-with-wiznet5100-chip
[7] https://github.com/knolleary/pubsubclient
[9] http://docs.openhab.org/addons/bindings.html
[10] http://docs.openhab.org/installation/openhabian.html
[11] https://sourceforge.net/projects/win32diskimager
[12] https://github.com/openhab/openhab1-addons/wiki/Explanation-of-items
[13] http://mqttfx.org
[14] https://github.com/sandro-k/MQTTLensChromeApp
[15] www.putty.org
[16] http://docs.openhab.org/configuration/persistence.html
[17] https://github.com/computourist/RFM69-MQTT-client
[18] https://github.com/computourist/ESP8266-MQTT-client
[19] http://homeautomation.proboards.com/board/2/openhab-rfm69-based-arduino

eLABS PROJECT

Dimmbare Außenbeleuchtung



INFOS ZUM PROJEKT

LED-Lampen | Dimmer | Terrassenbeleuchtung | Bewegungsmelder

Einsteiger
➡ Fortgeschrittener
Experte

Etwa 3 Stunden

FTDI-USB/Seriell-Kabel, PIC MikroBasic Demoversion, Visual Basic 2015 (alles optional)

Rund 25 Euro

Von **Andreas Meyer** (D)

Eines schönen Sommertages installierte sich unser Leser Andreas Meyer eine Terrassenbeleuchtung, bestehend aus vier 3-W-LED-Spots. Das Resultat war aber verdammt hell. Da niemand gerne an lauen Sommerabenden mit Sonnenbrille auf der Terrasse sitzt, wurde die neue Beleuchtung kaum genutzt. Also musste ein Dimmer her, der sanft auf niedrigere Helligkeit fährt und noch weiteren Luxus bietet...

Einstellbare Helligkeiten für 12-V-LED-Lampen

Nicht nur für Elektroniker gilt das folgende, eherne „Beleuchtungsgesetz“: *Wo eine Lampe viel zu hell, da muss ein Dimmer her, und schnell!*
Die vier LED-Spots von Andreas Meyer waren leider nicht von Hause aus für das Dimmen freigegeben. Allerdings kann das bei Exemplaren, die für eine 12-V-Versorgung ausgelegt sind, nichtsdestotrotz prima funktionieren (bei LED-Lampen für das Lichtnetz allerdings fast nie).

Ein kleiner Test mit PWM-Signal aus einem Mikrocontroller samt Stromverstärkung per Power-MOSFET (**Bild 1**) zeigte, dass sich seine Spots sehr wohl dimmen ließen, obwohl nichts Derartiges aufgedruckt war. Grund für das unterschiedliche Verhalten von 230-V- und 12-V-Exemplaren ist, dass die LED-Lampen für das Lichtnetz in der Regel ein kleines Schaltnetzteil enthalten, das aus den 230 V Wechselspannung einen geeig-

Eigenschaften

- Anschlussmöglichkeit von vier LED-Lampen
- Zwei Dimmstufen für Niedervolt-LED-Lampen
- Sanfter Übergang zwischen den Helligkeiten
- Steuerung der Dimmstufen über Bewegungsmelder

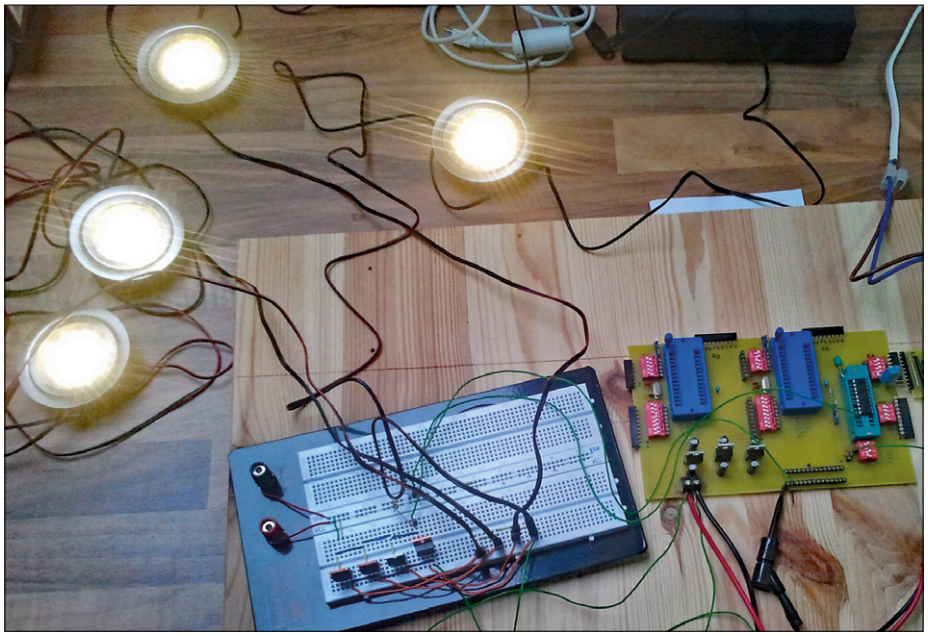


Bild 1. Der schnelle Testaufbau des Autors: Programmer mit Mikrocontroller und vier MOSFETs auf einem Steckbrett.

Perfektes Licht für laue Sommerabende!

neten Konstantstrom für die LEDs macht. Dieses Netzteil muss natürlich explizit für das Dimmen gebaut sein und den Ausgangsstrom proportional zur mittleren Eingangsspannung reduzieren. Andernfalls versucht das Netzteil, den Konstantstrom solange es geht stabil zu halten; und wenn es nicht mehr geht, wird es dunkel, flackert oder riecht verdächtig „nach Elektronik“. 12-V-Lampen hingegen sind sehr oft simpler gestrickt: Es sind häufig jeweils drei weiße LEDs in Serie geschaltet und mit einem passenden Vorwiderstand versehen, sodass an 12 V Gleichspannung ein halbwegs konstanter Strom fließt. Da hier keine Elektronik regelnd eingreift, ist Dimmen per schlichter Variation des Puls/Pausen-Verhältnisses kein Problem.
Soweit Ausgangslage und Theorie.

Schaltungskriterien

Wenn schon der fliegende Aufbau auf einem Mikrocontroller basiert, dann darf so ein Chip in der fertigen Schaltung nicht fehlen. Doch zunächst mussten die Anforderungen festgelegt werden: Aufgrund der Erfahrung mit der zwar neuen aber kaum genutzten Terrassenbeleuchtung war klar, dass die Helligkeit deutlich niedriger als „volle Pulle“ sein musste. Außerdem sollte die Beleuchtung von einem handelsüblichen Dämmerungsschalter und/oder normalen Netzschalter aktiviert werden, denn tagsüber sollte sie nicht leuchten und auch nachts nur, wenn gewünscht. Weiter sollte es kurzfristig heller werden, wenn sich jemand bewegt, denn beim Nachschubholen in

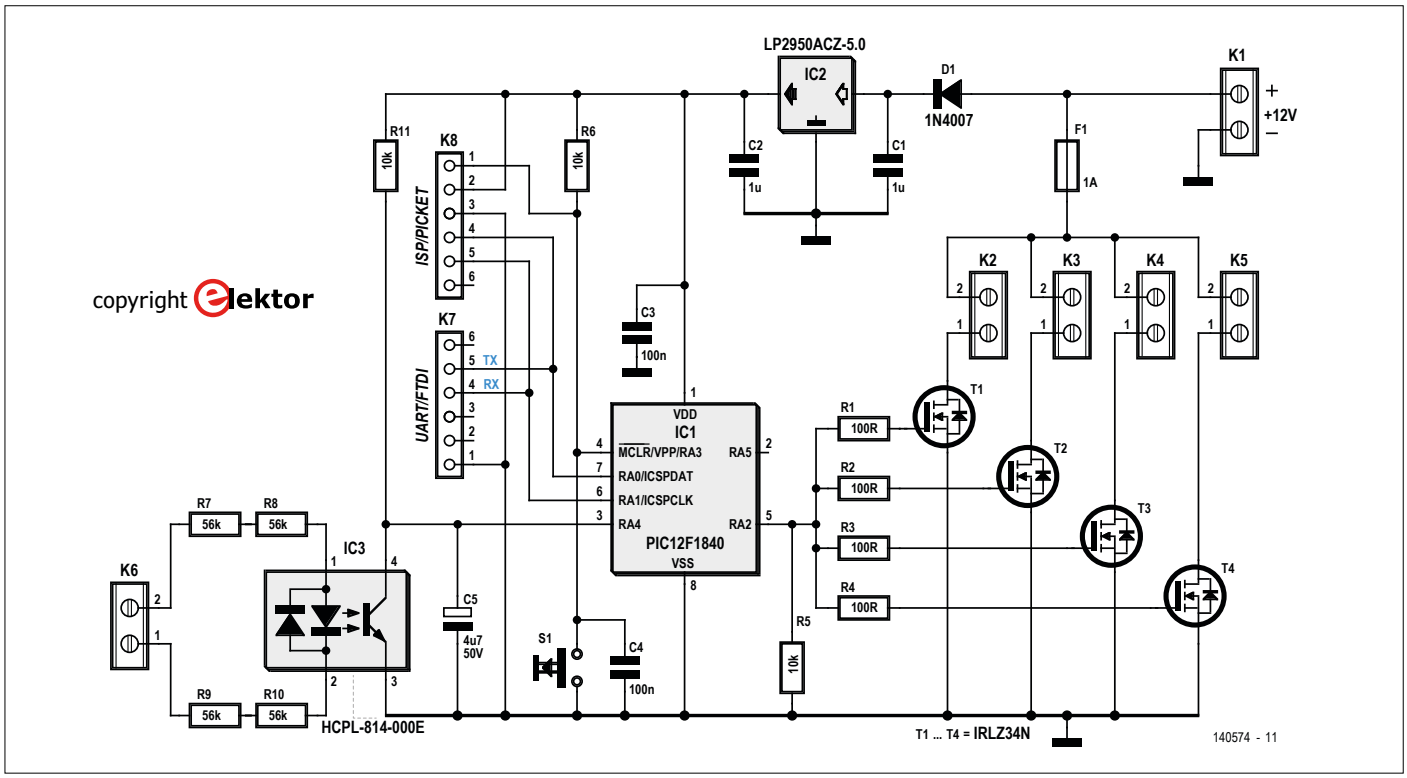


Bild 2. Die Schaltung der dimmbaren Außenbeleuchtung besteht aus dem Mikrocontroller, vier Power-MOSFETs, einem 230-V-Eingang für einen Bewegungsmelder und einem 5-V-Spannungsregler.

Form von Bier, Wein oder Wasser soll niemand stolpern. Last, but not least sollten die Helligkeitsstufen aus Komfortgründen nicht abrupt, sondern kontinuierlich geändert werden.
Das Dimmen selbst ist mit PWM per MCU sicherlich kein Problem. Dämmerungsschalter kauft man bei Bedarf und setzt sie einfach in Serie mit dem Lichtschalter. Für die Bewegungsdetektion fügt man einen geeigneten Bewegungsmelder ein – in jedem Baumarkt werden sie einem nachgeworfen. Ergo braucht die Elektronik keinen Helligkeitssensor, aber einen 230-V-Eingang für den Bewegungsmelder, damit die Beleuchtung zwischen den beiden gewünschten Helligkeitsstufen umschalten kann. Hier haben sich beim Autor übrigens PWM-Tastverhältnisse von 10 % für die normale Stimmungsbeleuchtung und von 50 % im Falle einer detektierten Bewegung als brauchbar herausgestellt. Die Elektronik brauchte vier PWM-Ausgänge und zudem ein Netzteil, das auch die 12-V-LED-Lampen versorgen kann. Wenn man für die Adaption der Helligkeitswerte nicht jedes Mal Konstanten im Quellcode ändern, kompilieren und die MCU neu flashen will, macht die nachträgliche Einstellung der PWM-Werte von außen via PC sehr viel Sinn, was eine serielle Schnittstelle erfordert. Damit dürften schon die wesentlichen Anforderungen an die Schaltung fixiert sein.

STÜCKLISTE

Widerstände:
(alle 5 %, ¼ W, 250 V)
R1..R4 = 100 Ω
R5,R6,R11 = 10 k
R7..R10 = 56 k *

Kondensatoren:
C1,C2 = 1 µ / 50 V, Folie, RM 2/10“
C3,C4 = 100 n / 50 V, Folie, RM 2/10“
C5 = 4µ7 / 35 V, Elko, stehend 5 x 11 mm, RM 2 mm

Halbleiter:
D1 = 1N4007
T1..T4 = IRLZ34NPBF
IC1 = PIC12F1840-I/P, programmiert *
IC2 = LP2950ACZ-5.0
IC3 = HCPL-814-000E, Optokoppler

Außerdem:
5x zweipolige Schraubklemmen für Platinenmontage, RM 2/10“
1x zweipolige Schraubklemme für Platinenmontage, RM 3/10“
2x 6-pol. Stiftleiste, einreihig, RM 1/10“
1x Sicherungshalter für Platinenmontage (z.B. Schurter 0031.8201)
1x Sicherung 1 A, 20 x 5 mm *
1x Taster für Platinenmontage, Schließer, 6 x 6 mm
1x Platine 140574-1 V2.2 *
1x 8-pol. IC-Fassung, DIP, für IC1

* siehe Text

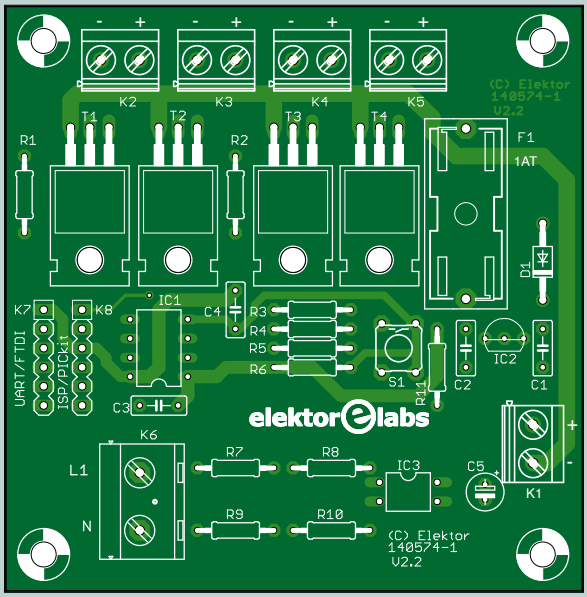


Bild 3. Die Platine für die dimmbare Außenbeleuchtung hat eine Kantenlänge von nur ca. 7,7 cm und kann in entsprechend große Verteilerdosen eingebaut werden.

Konkrete Schaltung

Zunächst baute Andreas Meyer die Schaltung rund um einen schon vorhandenen Mikrocontroller des Typs PIC16F628 auf und nutzte als 230-V-Steuereingang für den Bewegungsmelder einen nicht ganz empfindlichen Optokoppler. Aufgrund des Feedbacks auf Elektor-Labs [1] und der Überarbeitung im Elektor-Labor wurde der Controller in der finalen Version dann durch den voll ausreichenden und kleineren sowie preiswerteren Typ PIC12F1840 ersetzt (siehe **Bild 2**).
Die Schaltung ist schnell erklärt: Oben rechts liefert ein mit mindestens 1 A belastbares und stabilisiertes 12-V-Netzteil die Versorgungsspannung. Das eine Ampere reicht für die vier 3-W-Lampen aus. Will man kräftigere Lampen, muss man das Netzteil und die Sicherung F1 anpassen. Die vier Lampentreiber in Form von Power-MOSFETs verkraften selbst ungekühlt auch deutlich mehr Strom. Zur Versorgung des Mikrocontrollers ist mit IC2 ein „Micropower“-Spannungsregler mit 5 V vorgesehen. Hier fließen nur wenige mA. Über K6 gelangen 230 V über Vorwiderstände an einen empfindlichen Optokoppler, wenn der Bewegungsmelder anspricht. Nicht nur zur Verteilung der Verlustleistung (insgesamt rund 240 mW), sondern auch aufgrund der Spannungsfestigkeit sind hier vier 56-kΩ-Widerstände in Serie geschaltet. Am Ausgang von IC3 glättet C5 die Spannung und verhindert so 100-Hz-Nadelimpulse. Der interne UART der MCU ist nicht nur an K7 gelegt, an dessen Header direkt ein FTDI-USB/Seriell-Kabel angeschlossen werden kann, sondern auch an K8, worüber sich die MCU mit Software versorgen = programmieren lässt. Schließlich ist mit S1 auch noch ein Reset-Taster vorhanden.
Die PWM-Frequenz wurde von der Firmware auf 150 Hz festgelegt. Das reicht einerseits praktisch immer für flimmerfreies Dimmen und ermöglicht andererseits eine niedrige (interne) Taktfrequenz des Mikrocontrollers von nur 200 kHz für eine geringe Stromaufnahme. Wem die PWM-Frequenz zu niedrig sein sollte, der kann die Taktfrequenz in der Firmware höher wählen, sollte dabei aber auch die Zeitkonstanten für den Übergang der beiden Dimmstufen im Auge behalten, die ja ebenfalls von der Taktfrequenz abhängig sind.

Platine, Software und Co.

Die für diese Schaltung konzipierte Platine (**Bild 3**) passt mit 7,7 cm Kantenlänge in größere Kabelverzweiger-Dosen. Wie man sieht, wurde komplett auf SMDs verzichtet. Das Bestücken dürfte dank bedrahteter Bauteile also leicht fallen. Neben der Platine ist auch ein fertig programmierter Mikrocontroller direkt bei Elektor erhältlich [2]. In dessen Firmware sind die beiden Startwerte für die PWM von 10 % und 50 % vorgegeben, die sich aber noch nachträglich via PC mit einer eigenen App anpassen lassen. Für IC1 empfiehlt sich die Verwendung einer IC-Fassung, wie in der Stückliste vorgesehen. Wie schon erwähnt, müssen die vier

Bild 4. Fenster der PC-Anwendung, mit der man die beiden Dimmstufen und die Änderungsgeschwindigkeiten einstellen kann.

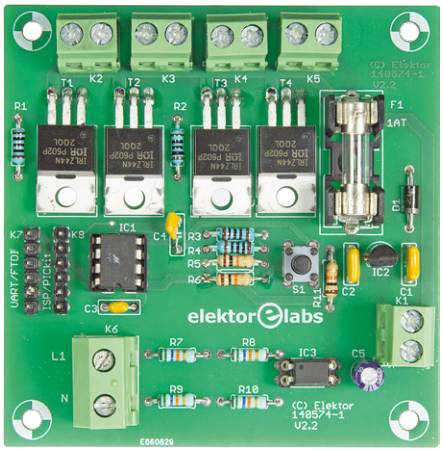


Bild 5. So sieht der im Elektor-Labor aufgebaute Prototyp aus.

MOSFETs nicht gekühlt werden. Bei einem typischen On-Kanalwiderstand von 35 mΩ fällt an jedem Exemplar selbst bei einem Strom von 1 A nur eine Verlustleistung von 35 mW an, und 1 A reichen sogar für 4 x 12 W, was sehr viel Licht ergibt, wenn man es mit LEDs erzeugt. Man muss übrigens auch nur so viele MOSFETs bestücken, wie man Ausgangskanäle benötigt. Schließt man z.B. nur zwei Lampen an, reichen auch zwei MOSFETs samt Vorwiderständen und Klemmen – es könnten dann also z.B. T3 und T4 samt R3 und R4 sowie K4 und K5 entfallen. Dass man F1 und das Netzteil bei einer eventuell höheren Gesamtleistung als 12 W anpassen muss, versteht sich von selbst.

Die Firmware des PIC12F1840 wurde mit der IDE mikroBasic von MikroElektronika entwickelt. Da sie aufgrund weniger Funktionen so klein ausfällt, kann sie sogar mit der kostenlosen Demoversion dieser IDE bearbeitet und geändert werden. Darüber hinaus hat Andreas Meyer mit Hilfe von Visual Studio eine kleine PC-App für Windows entwickelt (siehe **Bild 4**), mit der sich die beiden PWM-Werte für die beiden Helligkeiten über ein FTDI-USB/Seriell-Konverterkabel an den Mikrocontroller übertragen lassen. Noch ein Hinweis für eigene Änderungen: Neue Werte werden von der MCU erst nach einem Reset übernommen! Der gesamte Source-Code, das fertige PC-Programm und die Hex-Datei der Firmware stehen selbstverständlich auf der Elektor-Webseite zu diesem Projekt zum kostenlosen Download zur Verfügung [2]. **Bild 5** zeigt den im Elektor-Labor aufgebauten Prototypen.

Noch etwas...

Man sollte definitiv nicht auf die Idee kommen, die Elektronik mit einem dieser preiswerten „elektronischen 12-V-Trafos" im länglichen weißen Plastikgehäuse zu versorgen, wie sie z.B. für Halogenlampen angeboten werden. Dabei handelt es sich in der Regel um einfache Schaltnetzteile, die sekundärseitig zwar 12 V bieten, aber dies als Wechselspannung mit einigen „zig" kHz. So eine Versorgung ist tatsächlich nur für Glühlampen geeignet und würde selbst direkt angeschlossenen LED-Lampen (auch ohne die hier beschriebene Elektronik) nicht gut bekommen. Die MOSFETs haben ja alle eine parasitäre, antiparallel geschaltete Diode integriert, wie man am Schaltsymbol sehen kann. Die angeschlossenen LED-Lampen bekämen so also immer mindestens eine Halbwelle der Wechselspannung ab, und wenn dann beim auf 50...100 % reduzierten Dimmbereich wirklich Wechselspannung an den LEDs liegt, steigert das nicht unbedingt deren Lebensdauer, da LEDs nur für geringe Sperrspannungen gebaut sind.

Der zweite zu beachtende Punkt betrifft die Installation selbst: An den vier Klemmen K2...K5 liegt bei angeschlossenen Lampen eine Rechteckspannung an, deren Frequenz der PWM entspricht. 150 Hz klingt zwar nach nicht viel, doch die Rechteckspannung hat ziemlich steile Flanken und dies entspricht einem hohen Anteil an Oberwellen, die durchaus in den klassischen HF-Bereich hineinragen. Mit anderen Worten: Wenn die Leitungen zu den LED-Lampen nicht recht kurz ausfallen, wird HF-Störstrahlung in den Äther und zur Nachbarschaft geschickt. Übliche Hausinstallationen mit normalen Kabeln oder Einzelleitern aus Starrdraht sind nicht für HF gemacht, sind also nicht „HF-dicht". In solchen Fällen ist verdrillter Klingeldraht ein Stück besser und noch besser ist bei Leitungen von über ein paar Metern Länge die Verwendung von abgeschirmten Kabeln. Sowohl Klingeldraht als auch normale RG-58/U-Koaxialkabel vertragen locker 2 A und sind vielleicht nicht für diesen Zweck vorgesehen, haben aber bei einer solchen Niedervolt-Verkabelung eine geringere Störabstrahlung.

Wenn Sie die dimmbare Außenbeleuchtung so wie Andreas Meyer in Kombination mit einem Bewegungsmelder (also mit zwei Dimmstufen) einsetzen, werden Sie je nach Lampeninstallation vermutlich mehrmals die beiden Helligkeitswerte anpassen. Es lohnt sich daher, die Schaltung noch nicht fertig einzubauen, bis Sie den optimalen Kompromiss an abendlicher Beleuchtung gefunden haben – zumindest ein leichter Zugang zu K7 für den Anschluss an einen PC oder Laptop (oder zu K8, falls Sie die Firmware mit geänderten Werten neu einspielen wollen) erleichtert die Optimierung beträchtlich. Und dann ist alles bereit für laue Sommerabende...

(140574)

Weblinks

[1] http://goo.gl/izezWr

[2] www.elektormagazine.de/140574

Über den Autor:

Andreas Meyer ist Industrieelektroniker und begeistert sich für die Programmierung von Mikrocontrollern und das Entwerfen von Schaltungen und Platinen-Layouts. Aktuell beschäftigt er sich vor allem mit der Programmierung von Windows-Anwendungen, die Daten mit Mikrocontrollern austauschen und visualisieren.

EINKAUFSZETTEL

→ 140574-1
Platine

→ 140574-41
Programmierter Controller

Anwesenheitssimulator

Vorbeugen gegen Einbruch

Von **Theo de Wijs** (NL)

Haus- und Wohnungseinbrüche sind nicht selten. Die Diebe begehen ihre Straftaten meistens dann, wenn die Bewohner nicht im Haus sind. Dann fühlen sich die kriminellen Elemente unbeobachtet und ungestört. Doch wenn Indizien auf die rechtmäßige Anwesenheit von Personen hindeuten, lassen die meisten zwielichtigen Gestalten von ihrem Vorhaben ab.

Die wirkungsvollste Vorkehr gegen Einbruch besteht vermutlich darin, den zu schützenden Bereich niemals zu verlassen. Aus naheliegenden Gründen ist dies kaum realistisch. Eine längere Abwesenheit, zum Beispiel während des Urlaubs, steigert die Gefahr, Opfer einer solchen Straftat zu werden. Für den Autor war dies Anlass genug, nach einem Weg zu suchen, der daran etwas ändert. Eine naheliegende Option ist die Installation einer zuverlässigen Alarmanlage. Doch dieses Vorhaben ist stets mit mehr oder weniger hohen Investitionen verbunden. Baumärkte und Discounter haben bisweilen preiswerte Alarmsysteme im Angebot, allerdings wird diesen Produkten nicht selten mangelnde Qualität bescheinigt. Außerdem ist es zu dem Zeitpunkt, in dem der Alarm auslöst, meistens schon zu spät.

Ein anderer Weg ist das permanente Simulieren der Anwesenheit von Bewohnern. Dazu genügt sicher nicht der Einsatz einer simplen Zeitschaltuhr, die zu stets gleichen Zeiten die Beleuchtung einschaltet. Auch Kriminelle mit nur mittlerer Intelligenz lassen sich damit kaum abschrecken. Der hier vorgestellte Anwesenheitssimulator wird vom Umgebungslicht gesteuert, wenn er abends die Rollläden herablässt. Erst kurze Zeit später werden die Beleuchtung und ein TV-Simulator eingeschaltet, und zur typischen Schlafengehzeit wird beides abgeschaltet. Am nächsten Morgen sorgt der Simulator dafür, dass die Rollläden hochgezogen werden.

Die Schaltung

In der Schaltung, die **Bild 1** zeigt, sind die Funktionsdetails unschwer zu erkennen. Die Betriebsspannung wird über C1 und R1 links im Bild aus der Netzspannung abgeleitet. Die Dioden D1...D4 richten die Spannung gleich, und D5 zusammen mit C2 formen daraus die Gleichspannung 12 V. **Achtung: Die gesamte Schaltung ist unmittelbar mit dem Stromnetz verbunden!** Das bedeutet nicht nur, dass sie absolut berührsicher

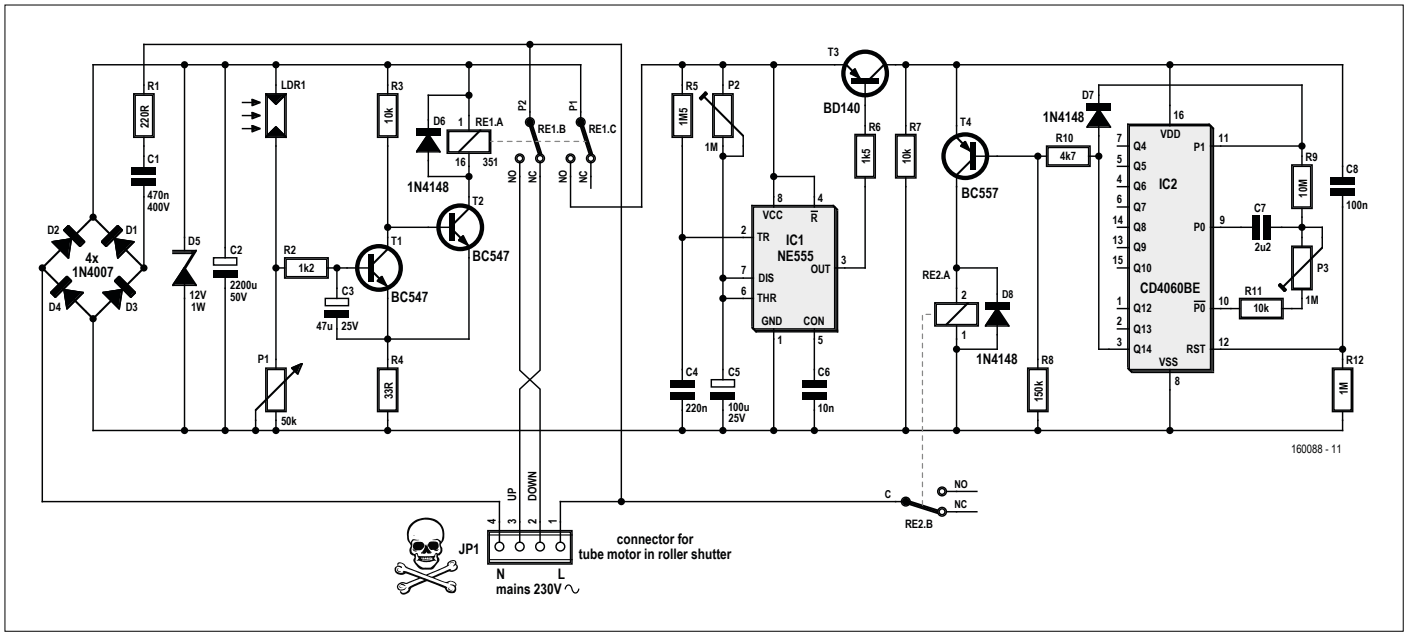


Bild 1. Die Schaltung steuert über zwei Relais die Rollladen, die Beleuchtung und einen TV-Simulator.

aufgebaut werden muss, sondern auch, dass Messungen ebenso wie versuchsweise Modifikationen **lebensgefährlich** sein können!

An die Stromversorgung schließt sich der lichtabhängige Spannungsteiler mit LDR1 und P1 an. Solange tagsüber genügend Licht auf den LDR fällt, ist der Widerstand niedrig. Die Spannung am Verbindungspunkt mit P1 ist folglich hoch. Transistor T1 leitet, und an R3 fällt Spannung ab. Das hat zur Folge, dass T2 sperrt und durch Relais RE1 kein Strom fließt. Der Arbeitskontakt des zweiten Umschaltkontakts (RE1.C) ist offen, die Schaltung befindet sich im Ruhezustand. Über den Ruhekontakt des ersten Umschaltkontakts (RE1.B) gelangt Netzspannung zum Kontakt UP von Anschluss JP1. Dieser Anschluss steuert den Rohrmotor des Rollladens, der den Rollladen hochzieht. Wenn der Rollladen hochgezogen ist, bleibt er in dieser Stellung stehen. Solange dieser Betriebszustand andauert, ist die übrige Schaltung mit IC1 und IC2 *nicht* mit der Betriebsspannung verbunden. Während der Dämmerung steigt der Widerstand des LDR, so dass die Spannung am Verbindungspunkt mit P1 sinkt. Mit fortschreitender Dunkelheit nimmt die Spannung so weit ab, dass T1 zu sperren beginnt. Die Lichtintensität, bei der dies geschieht, ist mit P1 einstellbar. Wenn T1 sperrt, steuert T2 durch, so dass das Relais anzieht. Damit ist der Anwesenheitssimulator aktiviert. Über den Arbeitskontakt des ersten Relais-Umschaltkontakts (RE1.B) gelangt nun Netzspannung zum Kontakt DOWN des Anschlusses JP1. An die zugehörige Leitung des Rollladen-Rohrmotors wird jetzt Netzspannung gelegt, der Rollladen wird herabgelassen.

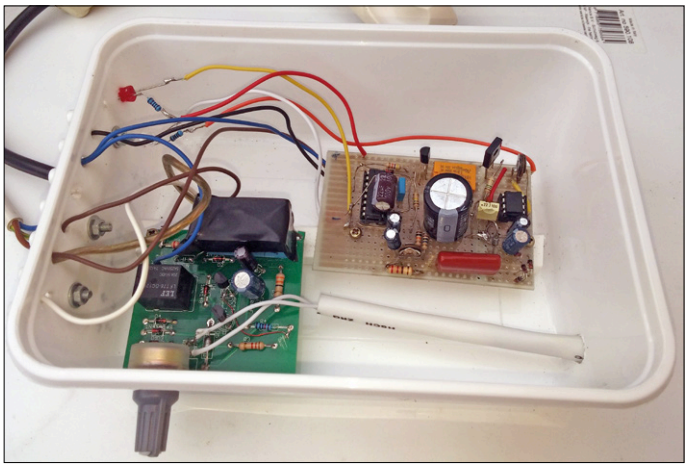


Bild 2. Der Musteraufbau des Autors ist konstruktiv und gestalterisch noch entwicklungsfähig...

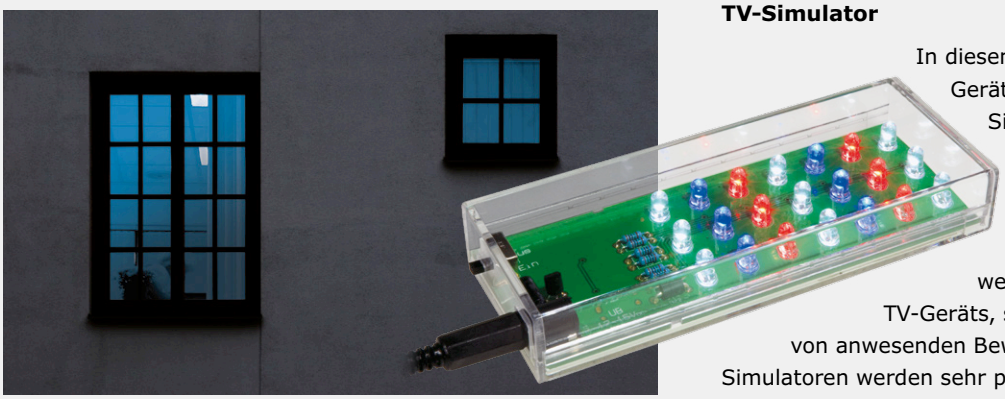
Die Timer

Über den Arbeitskontakt des zweiten Relais-Umschaltkontakts (RE1.C) gelangt die Betriebsspannung nun auch zum übrigen Teil der Schaltung. Timer IC1, ein altbewährter 555, hat eine wichtige Funktion. Wenn gleichzeitig mit dem Herunterlassen des Rollladens die Beleuchtung oder ein anderes von draußen erkennbares Gerät eingeschaltet wird, kann ein potentieller Täter Verdacht schöpfen. Er kann auf die Idee kommen, dass beide Ereignisse parallel von einer Schaltuhr gesteuert werden und die Bewohner nicht wirklich anwesend sind. Damit wäre die abschreckende Wirkung des Simulators zumindest in Frage gestellt.

In dem Moment, in dem Timer 1 Betriebsspannung erhält, ist sein Ausgang (Pin 3) auf hoher Spannung. Transistor T3 sperrt, so dass die Betriebsspannung des zweiten Timers (IC2) unterbrochen ist. Nach Ablauf der mit Trimmpoti P2 einstellbaren Kippdauer von IC1 schaltet T3

Über den Autor

Theo de Wijs (73) war leitender Mitarbeiter einer bekannten Hilfsorganisation. Die Ausbildung im Bereich der Elektronik absolvierte er zu einer Zeit, als die Geräte noch von Röhren und ersten Transistoren dominiert wurden. Bis 1976 war die Elektronik das Betätigungsfeld (unter anderem bei Philips), dann folgte der Ruf zu der neuen Aufgabe. Das Interesse für die Elektronik hat trotzdem nie nachgelassen.



TV-Simulator

In diesem Beitrag wird wiederholt ein Gerät mit der Bezeichnung „TV-Simulator“ erwähnt. Das ist ein simples kleines Gerät, aufgebaut mit einigen farbigen LEDs und einem (Pseudo-)Zufallsgenerator. Die Elektronik simuliert das wechselnde Leuchten eines Farb-TV-Geräts, so dass ein ferner Beobachter von anwesenden Bewohnern ausgehen muss. TV-Simulatoren werden sehr preiswert im Handel angeboten.

die Betriebsspannung von IC2 ein.
IC2, ein 4066, ist ein binärer Zähler, er zählt die Impulse des eingebauten Oszillators. Die Zählerstufen haben diverse Ausgänge, sie sind mit Q4...Q14 bezeichnet. Beim Reset mit C8 und R12 an Eingang 12 sind sämtliche Ausgänge einschließlich Q14 (Pin 3) zunächst auf niedriger Spannung. Das hat zur Folge, dass PNP-Transistor T4 leitet und Relais RE2 anzieht. Der Arbeitskontakt von RE2 schaltet die Beleuchtung, den TV-Simulator oder ein anderes auffälliges Gerät ein. Nach Ablauf der mit P3 in weitem Bereich einstellbaren Zeit (bis etwa 24 Stunden) geht Ausgang Q14 auf hohe Spannung. T4 sperrt, und das Relais RE2 fällt ab. Dies geschieht zu der simulierten Schlafengehzeit, die Beleuchtung und der TV-Simulator werden abgeschaltet. Jetzt blockiert Diode D7 den internen Oszillator des Timers IC2, so dass der Zählvorgang stoppt.
Damit ist der Anwesenheitssimulator in den Ruhezustand zurückgekehrt: Zähler IC2 zählt nicht mehr, und Relais RE2 ist abgefallen. Dieser Zustand bleibt so lange erhalten, bis die Sonne aufgeht und das Tageslicht anbricht. Dann fällt mehr Licht auf LDR1, die Spannung am Verbindungspunkt des LDR mit Poti P1 steigt und Transistor T1 beginnt zu leiten. Da nun T2 sperrt, fällt Relais RE1 ab, die Schaltung befindet sich wieder im Ruhezustand. Jetzt ist auch die Betriebsspannung von IC1 und IC2 abgeschaltet, so dass bei Anbruch der Dunkelheit ein Reset stattfindet und ein neuer Zyklus beginnt.

Weitere Details

Die Schaltung ist so konzipiert, dass beide Relais aktiviert sind, solange es draußen dunkel ist. Das ist naheliegend, denn meistens findet der Jahresurlaub im Sommer statt, wenn die Nächte kürzer als die Tage sind. Falls der Simulator während der dunklen Jahreszeit zum Einsatz kommen soll (Wintersporturlaub!), kann T2 gegen einen PNP-Transistor getauscht werden. Die Anschlüsse der Relaiskontakte sind dann ebenfalls gegeneinander zu tauschen. In diesem Fall ist das Relais bei Dunkelheit aktiviert, was sogar einen etwas niedrigeren Strombedarf zur Folge hat.
Der Rohrmotor, der den Rollladen herablässt und hochzieht, muss unbedingt mit so genannten Endstoppkontakten ausgestattet sein. Die Kontakte sorgen dafür, dass der Motor in den Endstellungen (vollständig aufgerollt, vollständig herabgelassen) selbsttätig abschaltet. Die meisten Rohrmotor-Typen besitzen diese Fähigkeit.
Die Relaiskontakte müssen den zu schaltenden Strömen und Spannungen gewachsen sein. Während Relais RE2 nur für die Beleuchtung und den TV-Simulator zuständig ist, muss RE1 mit einem Umschaltkontakt den Rohrmotor schalten. Für RE1 wird deshalb ein Typ empfohlen, der 10 A oder höher schalten kann.

Wir haben schon darauf hingewiesen, dass die Schaltung ohne Netztrafo arbeitet und unmittelbar mit dem Stromnetz in Verbindung steht. Das ist nur dann ungefährlich, wenn dem Aufbau und dem Einsatz die gebotene Vorsicht und Sorgfalt zuteilwerden. Die Schaltung muss ihren Platz in einem stabilen, absolut berührsicheren Kunststoffgehäuse haben. Einer besonderen Aufmerksamkeit bedarf das Potentiometer P1: Hier darf nur eine Ausführung mit isolierender Kunststoffachse Verwendung finden!

Bild 2 zeigt den Versuchsaufbau des Anwesenheitssimulators beim Autor. Die teilweise auf einer Lötpunktraster-Platine realisierte Schaltung ist als *Proof-of-concept*-Konstruktion zu verstehen. Trotzdem erfüllt sie in dieser Variante schon einige Zeit ihren Zweck. Beispielhaft für den Aufbau einer Schaltung, die über Wochen unbeaufsichtigt ihren Dienst tun soll, kann diese Konstruktion nicht sein. ◀

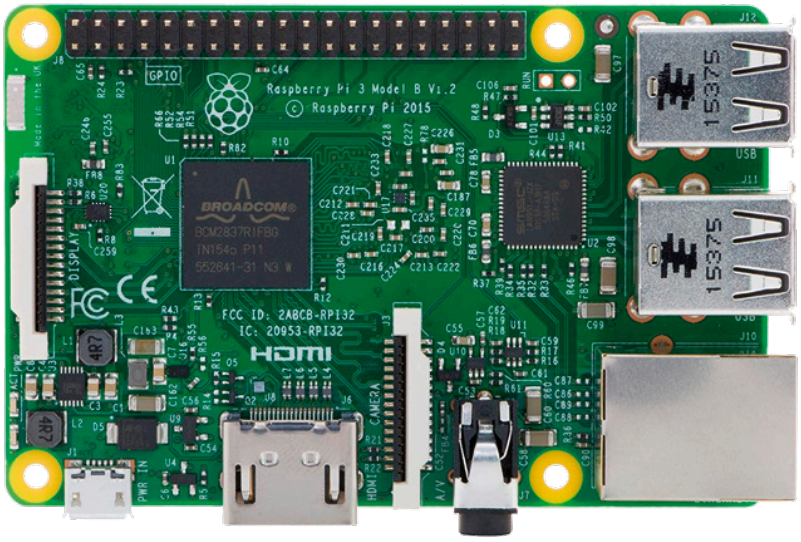
(160088)gd

Rohrmotoren

Ein Rohrmotor ist ein langgestreckter Elektromotor in einem rohrförmigen Gehäuse. Ein typischer Einsatzbereich ist der Antrieb von Rollläden und Markisen. Der Rohrmotor befindet sich dort in der hohlen Aufwickelachse. Mit der Aufwickelachse wird der Rohrmotor mechanisch über einen Flansch und einen Mitnehmer gekoppelt.
In den technischen Beschreibungen sind stets zwei Größen angegeben: Das Drehmoment in Nm und die Drehgeschwindigkeit in U/min. Ein Rohrmotor 6/17 hat das Drehmoment 6 Nm und die Drehgeschwindigkeit 17 Umdrehungen in der Minute (*Quelle: Wikipedia*).

Android auf dem Raspi (2)

Display mit SPI ansteuern



In der letzten Ausgabe haben wir ein erstes Android-Programm zum Laufen gebracht: In einer Endlosschleife wurde ein Ausgang hoch- und runtergeschaltet, um das Echtzeitverhalten zu untersuchen [1]. Wenn wir Peripherie ansteuern wollen, ist das „manuelle“ Bedienen der Pins nicht sinnvoll; dann sollte man die Hardware-Bus-Interfaces des Controllers nutzen. In diesem Artikel zeigen wir die Ansteuerung eines kleinen OLED-Displays per SPI.

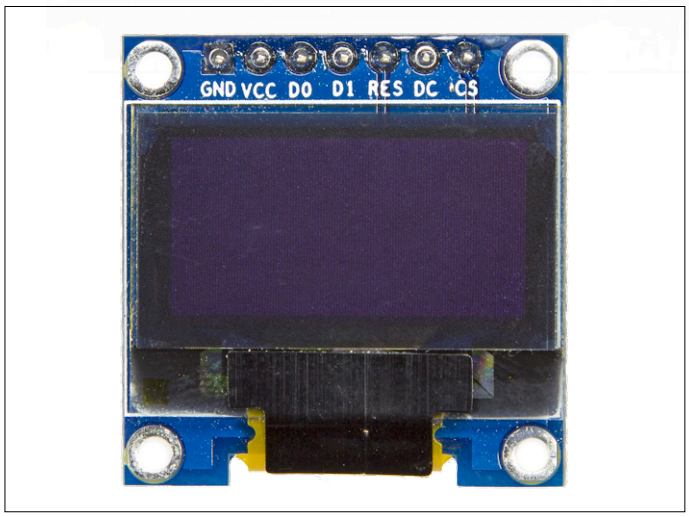


Bild 1. Dieses Display ist - sowohl vom Strombedarf als auch von der Größe her - wesentlich kompakter als ein HD-Monitor.

Von **Tam Hanna** (Slowakei)

Will man Peripherie mit Android Things ansteuern, hat man im Moment die Qual der Wahl zwischen SPI und I2C. Wir wollen in den folgenden Schritten eines der weit verbreiteten SPI-OLED-Displays verwenden: Bei AliExpress ist das in **Bild 1** gezeigte Teil schon für wenige Dollar verfügbar.

Schaltung

Ein Raspberry Pi mit Android verhält sich aus Hardwaresicht wie ein gewöhnlicher Raspberry Pi. Die Schaltung, die in **Bild 2** gezeigt wird, haben wir ganz ähnlich in der Serie „Windows auf dem Raspi“ [2] benutzt.

Android Things identifiziert SPI-Busse (genauso wie GPIO-Pins) durch Strings. Unsere erste Aufgabe besteht darin, die am Raspberry Pi zur Verfügung stehenden zu identifizieren:

```
@Override
protected void onCreate(Bundle savedInstanceState) {

    . . .

    PeripheralManagerService manager = new
         PeripheralManagerService();
    List<String> deviceList = manager.getSpiBusList();

    if (deviceList.isEmpty()) { } else {
        Log.i(TAG, "List of available devices: " +
              deviceList);
    }
}
```

Das Programm zeigt eine Liste aller verfügbaren Busse in der Debugger-Konsole. Zum Zeitpunkt der Drucklegung gibt der Raspberry Pi 3 des Autors zwei Endpunkte aus:

```
I/MainActivity: List of available devices: [SPI0.0,
      SPI0.1]
```

Gradle bzw. Android-Studio stellen an dieser Stelle eine kleine Falle: Wenn Sie die Aberntung der Daten über einen Breakpoint durchführen wollen, müssen Sie darauf achten, das Programm durch Anklicken des Debug-Symbols zu starten. Wenn Sie auf den normalen Run-Pfeil klicken, wird der Debugger nicht mit dem Programm verbunden.

In einem ersten Demo-Projekt (der Code kann natürlich wieder von der Elektor-Website heruntergeladen werden [3]) müssen wir zuerst den SPI-Bus initialisieren. Dazu wollen wir eine neue Methode namens `configureAndFindDevice` benutzen. Als Erstes erweitern wir die Haupt-Activity um ein globales GPIO-Objekt, das für die Steuerung der Reset-Leitung zum Display verantwortlich ist. Zudem legen wir ein SPI-Device an, das die eigentliche Kommunikation zwischen Display und Raspi kapselt.

```
public class MainActivity extends Activity {

    SpiDevice myDevice;

    Gpio myPinReset;
```

Im nächsten Schritt folgt die - mehr oder weniger klassische - Initialisierung des SPI-Busses innerhalb von `configureAndFindDevice` (**Listing 1**). Der Aufruf von `setFrequency` legt die Kommunikationsfrequenz fest – in Tests des Autors funktionierten sowohl 4 MHz als auch 8 MHz ohne Probleme.

Aufmerksame Leser des letzten Teils fragen sich mitunter, was `Thread.Sleep` im GUI-Thread zu suchen hat: Die Antwort darauf ist, dass eine Verzögerung von 10 ms nicht ins Gewicht fällt und dementsprechend den Aufwand zur Erzeugung eines eigenen Threads in keiner Weise rechtfertigt.

Den Aufruf der Methode bringen wir in `onCreate` unter, so dass er zum Programmstart erfolgt (**Listing 2**).

Hier initialisieren wir einen weiteren GPIO-Pin, der uns das Steuern der Data-Control-Leitung ermöglicht. Außerdem rufen wir die Funktion `initDisplay` auf, die für das Setzen diverser Parameter des Displays zuständig ist – der Inhalt dieser Methode folgt im nächsten Abschnitt.

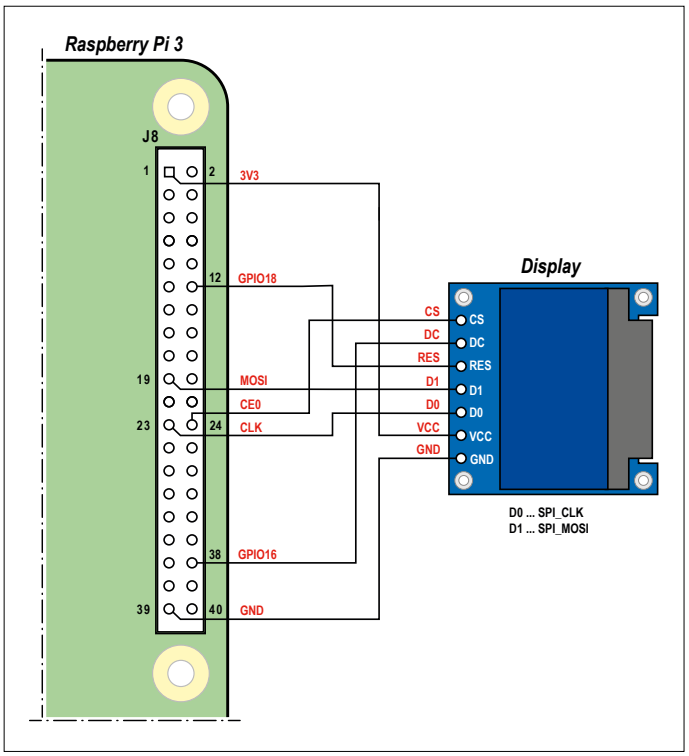


Bild 2. Einige Leitungen verbinden den Raspi und das Displaymodul.

Im Interesse der didaktischen Sauberkeit sei angemerkt, dass das SPI-Device beim Beenden der Activity freigegeben werden sollte – dazu dient Code in `onDestroy`, siehe **Listing 3**.

Listing 1. Initialisieren der SPI-Schnittstelle.

```
void configureAndFindDevice(){

    PeripheralManagerService manager = new
                PeripheralManagerService();

    try {

        myPinReset = manager.openGpio("BCM18");
        myPinReset.setDirection(Gpio.
            DIRECTION_OUT_INITIALLY_LOW);
        myPinReset.setValue(false);

        myDevice = manager.openSpiDevice("SPI0.0");
        myDevice.setMode(SpiDevice.MODE0);
        myDevice.setFrequency(4000000);
        myDevice.setBitsPerWord(8);
        myDevice.setBitJustification(false);
        Thread.sleep(10);
        myPinReset.setValue(true);
    }

 catch (Exception e) {. . .}

}
```

Listing 2. Die Initialisierungen aller Pins kommen in die Methode onCreate, die beim Programmstart aufgerufen wird.

```
@Override
protected void onCreate(Bundle savedInstanceState) {

    . . .

    configureAndFindDevice();

    try {

        PeripheralManagerService manager = new
                PeripheralManagerService();
        myPinDataCommand = manager.
                openGpio("BCM16");
        myPinDataCommand.setDirection(Gpio.
                DIRECTION_OUT_INITIALLY_LOW);

    }catch(Exception x){}

    initDisplay();

}
```

Fehlen diese Close-Kommandos, so kommt es beim Debugging mitunter zu Exceptions der Bauart „`android.os.ServiceSpecificException: BCM18 is already in use`". Beachten Sie zudem, dass Activity-Klassen vom Android-OS auch im laufenden Betrieb abgetragen werden dürfen - es handelt sich dabei um eine Design-Entscheidung der Entwickler, die in der Android-Welt an vielen Stellen für Unruhe sorgt.

Listing 3. Am Ende des Programms wird aufgeräumt.

```
@Override

protected void onDestroy() {

    super.onDestroy();

    Log.d(TAG, "onDestroy");

    try {
        if(myDevice!=null) myDevice.close();
        if(myPinDataCommand!=null) myPinDataCommand.
              close();
        if(myPinReset!=null) myPinReset.close();
        myDevice = null;

    } catch (IOException e) {

        Log.w(TAG, "Unable to close SPI device", e);

    }

}
```

Wir initialisieren ...

Nach dem erfolgreichen Konfigurieren des SPI-Frameworks können wir uns der Übertragung von Daten zuwenden, um den Displaycontroller in den korrekten Modus zu versetzen und Informationen auf den Bildschirm zu bringen.
Auch wenn das Datenblatt des SSD1306 weit verbreitet ist - es lohnt sich beim erstmaligen Einsetzen eines ICs immer, bei Adafruit und in der Dokumentation diverser anderer Hersteller nach fertigen Bibliotheken zu suchen, welche die Vorgehensweise demonstrieren.
Den Code von InitDisplay sieht man in **Listing 4**.

Zunächst erfolgt hier ein weiterer Aufruf von `Thread.sleep`: Er sorgt dafür, dass der Controller nach dem Aufwachen aus dem Resetzustand genügend Zeit zum Anlassen bekommt. Danach senden wir drei Kommandos; sie veranlassen, dass die Ladungspumpe angeworfen und das Display eingeschaltet wird. Vor dem Aufruf des für das Senden von Anzeige-Informationen zuständigen Kommandos `sendData` sorgen wir noch dafür, dass der Displaypuffer mehr oder weniger zufällige Informationen enthält.

Die eigentliche Deklaration der Kommando-Konstanten erfolgt nach einem etwas seltsamen Schema. Interessant ist hier vor allem, dass vor jeder einzelnen Konstante ein Cast auf Byte erforderlich ist. Fehlt dieser, so würde der Java-Compiler einen Baum aufstellen:

```
private byte[] CMD_DISPLAY_ON = { (byte)0xAF };
```

Als Nächstes können wir uns den beiden Methoden zuwenden, die Informationen in Richtung des Display-Controllers schicken. Hierzu eine kleine Anmerkung: Der SD1306 arbeitet sowohl beim Entgegennehmen von Daten als auch beim Einsammeln von Kommandos als gewöhnliches SPI-Gerät. Der Pegel am

Listing 4. Anzeige von Linien auf dem Display.

```
void initDisplay()

{

    try {

        Thread.sleep(100);
        SendCommand(CMD_CHARGEPUMP_ON);
        SendCommand(CMD_MEMADDRMODE);
        SendCommand(CMD_DISPLAY_ON);

        for(int i=0;i<myDisplayBuffer.length;i++){
            myDisplayBuffer[i]=22;
        }

        sendData();
    }

        catch(Exception x){Log.e("Elektor",x.
getMessage());}

}
```

digitalen DC-Eingang bestimmt den Modus. Ist das DC-Signal bei der Übertragung von Daten high, so wandern die Informationen in den Displaypuffer - ist es low, so werden die Informationen stattdessen in den Kommandospeicher geschrieben. Dies führt zu folgendem Code:

```
    void SendCommand(byte[] _what){

        try {
            myPinDataCommand.setValue(false);
            myDevice.write(_what, _what.length);
        }

        catch (Exception x){
            Log.e("Elektor",x.getMessage());
        }
    }
```

Die Version zur Übertragung von Anzeigedaten unterscheidet sich nur minimal von `SendCommand`. Anstatt die Daten aus dem Funktions-Parameter zu übernehmen, bedient sich die Funktion in einem als Member anzulegenden Array (`myDisplayBuffer`).

```
    void sendData(){

        try {
            myPinDataCommand.setValue(true);
            myDevice.write(myDisplayBuffer,
                 myDisplayBuffer.length);
        }

        catch (Exception x){
            Log.e("Elektor",x.getMessage());
        }
    }
}
```

Damit ist die erste Version des Programms auch schon einsatzbereit. Führen Sie es aus - bei korrekter Implementierung erscheint eine Serie von horizontalen Strichen auf dem Bildschirm (**Bild 3**).

... und kommunizieren

Zum Anzeigen von Linienpatterns ist kein dediziertes Echtzeitbetriebssystem erforderlich: Einen 1 KB großen Displayspeicher kann man auch auf einem AVR erfolgreich zumüllen. Die Leistungsfähigkeit des RPi können Sie nur dann ausspielen, wenn das Betriebssystem komplexe Informationen anzeigt.
Eine richtig interessante Aufgabe wäre das Anzeigen von GUI-Steuerelementen (zum Beispiel eines Diagramms), als Bitmap auf dem Display. Dazu wollen wir *MPAndroidChart* in unsere Lösung einbinden - es handelt sich dabei um eine hochleistungsfähige Diagrammbibliothek, die auch in der Entwicklung klassischer Android-Applikationen eine Rolle spielt. Öffnen Sie im ersten Schritt die Datei *build.gradle* des gesamten Projekts, und ergänzen Sie sie um folgende Passage:

```
allprojects {
    repositories {
        jcenter()
```

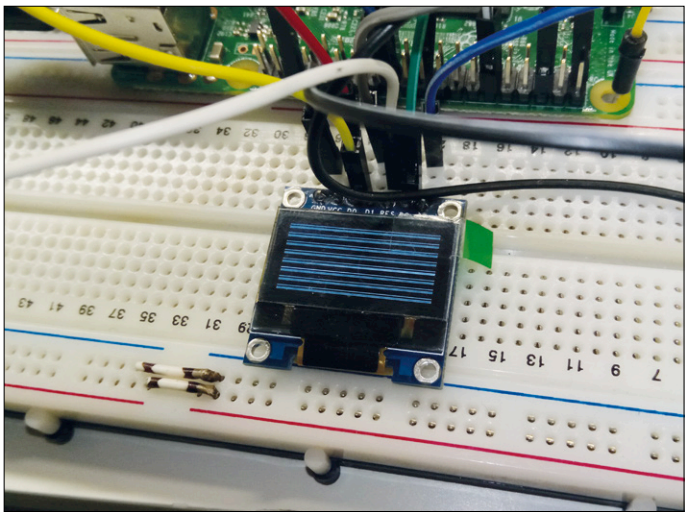


Bild 3. Das Testpattern besagt, dass alles problemlos funktioniert. Ein schwarzes OLED wirkt dagegen „tot“.

```
        maven { url "https://jitpack.io" }
    }
}
```

Die zur Applikation gehörende Datei *build.gradle (Module:app)* wird dagegen um Folgendes erweitert:

```
dependencies {
    provided 'com.google.android.
          things:androidthings:0.1-devpreview'
    compile 'com.github.PhilJay:MPAndroidChart:v3.0.1'
}
```

Nach dem Speichern dieser beiden Dateien blendet Android-Studio ein gelbes Banner ein, in dem Sie nun auf *Sync Now* klicken. Die IDE aktualisiert sodann das Projekt als Ganzes - eine der größten Stärken von Gradle ist seine Fähigkeit, Bibliotheken und sonstige zur Unterstützung benötigten Informationen während der Kompilation aus dem Internet herunterzuladen.
Im nächsten Schritt klicken Sie den Ordner des Projekts rechts an und wählen die Option *New -> File -> Android Resource File*. Im daraufhin erscheinenden Dialog wählen Sie die in **Bild 4** gezeigten Einstellungen aus, um ein Layout-File anzulegen.

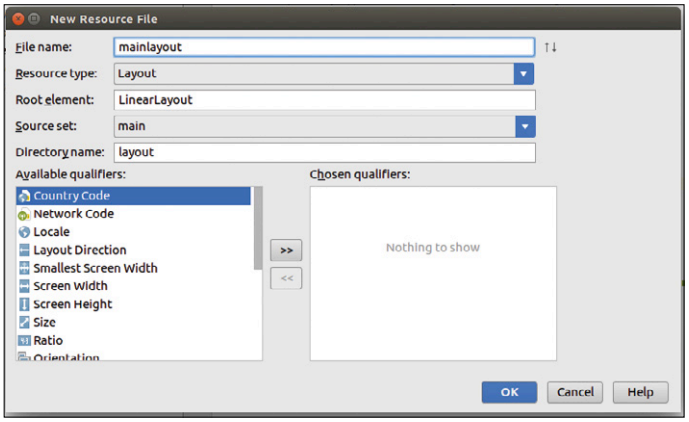


Bild 4. Diese Parameter erzeugen eine neue Layout-Datei.

Das in Android normalerweise implementierte Layout-Subsystem würde an dieser Stelle für Ärger sorgen, weil es die Größe des gezeigten Steuerelements dynamisch anpassen würde. Wir wollen hier aber ein 128 × 64 Pixel großes Widget haben, was folgende Änderungen im Code voraussetzt:

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android=http://schemas.android.com/
            apk/res/android
    android:orientation="vertical"
            android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.github.mikephil.charting.charts.LineChart
        android:id="@+id/chart1"
        android:layout_width="128px"
        android:layout_height="64px"/>

</LinearLayout>
```

Der hier verwendete Umweg über den Hauptbildschirm des Raspis ist nicht unbedingt notwendig, erleichtert aber die Arbeit, da wir die Resultate der Ausgabe auch auf einem angeschlossenen Studiomonitor ansehen können.
Im nächsten Schritt können wir uns der Chart-Bibliothek zuwenden, die naturgemäß vor der Anzeige von Diagrammen ein wenig Hilfe benötigt. Dies sieht folgendermaßen aus:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.mainlayout);
```

Der erste Unterschied zu gewöhnlichen Android-Things-Applikation ist der Aufruf von `setContentView` - er sorgt dafür, dass die im XML-Layout befindlichen Informationen geladen werden.
Sodann erstellen wir ein Array, das mehr oder weniger zufällige Anzeigeinformationen anliefert:

```
    ArrayList<Entry> values = new ArrayList<Entry>();

    for (int i = 0; i < 20; i++) {
        float val = (float) (Math.random() * 25) + 3;
        values.add(new Entry(i, val));
    }
```

Als Nächstes folgt die Parametrierung: MPAndroidChart ist für das Handhaben größerer Datenmengen vorgesehen, weshalb die Initialisierung relativ aufwändig ausfällt, was wir hier nur in Ausschnitten zeigen:

```
    LineChart mChart = (LineChart) findViewById(R.
         id.chart1);
    mChart.getDescription().setEnabled(false);
    LineDataSet s1=new LineDataSet(values,"");

    . . .

    ArrayList<ILineDataSet> dataSets = new
          ArrayList<ILineDataSet>();
    dataSets.add(s1);
    LineData data = new LineData(dataSets);
    mChart.setData(data);
```

Das primäre Problem, das zur enormen Länge des Codes führt, ist das Eliminieren von unnötigen Zeichenbefehlen - die Display-Auflösung von 128 × 64 Pixeln ist so klein, dass sie von den MPAndroidChart-Entwicklern wahrscheinlich nie getestet wurde.

Unsere nächste Aufgabe besteht nun darin, das Resultat in ein Bitmap umzuwandeln, das sodann den Weg zum Display antreten kann. Hier ist vor allem wichtig, dass die Farbinformationen verworfen werden müssen - unser Bildschirm ist ja schwarz-weiß-only.
Aufgrund des vergleichsweise seltsamen Zeichenverfahrens von MPAndroidChart müssen wir die Umwandlung verzögert ausführen. Dies lässt sich über einen Handler bewerkstelligen. Das Codeskelett für den Handler lautet wie folgt:

```
final Handler handler = new Handler();
handler.postDelayed(new Runnable() {

    @Override
    public void run() {

        //Code ist hier

    }
}, 700);
```

`postDelayed` nimmt ein Objekt entgegen, das eine `run()`-Methode bereitstellen muss. Hierin kommt der Code, der verzögert ausgeführt werden soll. Dieser ist sehr umfangreich – beginnen wir mit der Konversion des Widgets in eine Instanz der Bitmap-Klasse:

```
    public void run() {
        LineChart mChart = (LineChart) findViewById(R.
                id.chart1);

        Bitmap returnedBitmap = Bitmap.createBitmap(128,
                 64,Bitmap.Config.ARGB_8888);
        Canvas canvas = new Canvas(returnedBitmap);

        Drawable bgDrawable =mChart.getBackground();
        if (bgDrawable!=null)
            bgDrawable.draw(canvas);
        else
            canvas.drawColor(Color.WHITE);

        mChart.draw(canvas)
```

An dieser Stelle enthält `returnedBitmap` eine vollfarbige Version des von MPAndroidChart gerenderten Diagramms. Unsere nächste Aufgabe besteht darin, dieses Bitmap in einen Framebuffer umzuwandeln. Als erstes Ärgernis präsentiert sich hier der Aufbau des Bildschirmspeichers, der in eine Gruppe von „Pages“ aufgesplittet ist.

Der einfachste Weg zur Lösung derartiger Probleme ist, mit for-Schleifen die Speicherzellen Schritt für Schritt abzugrasen.

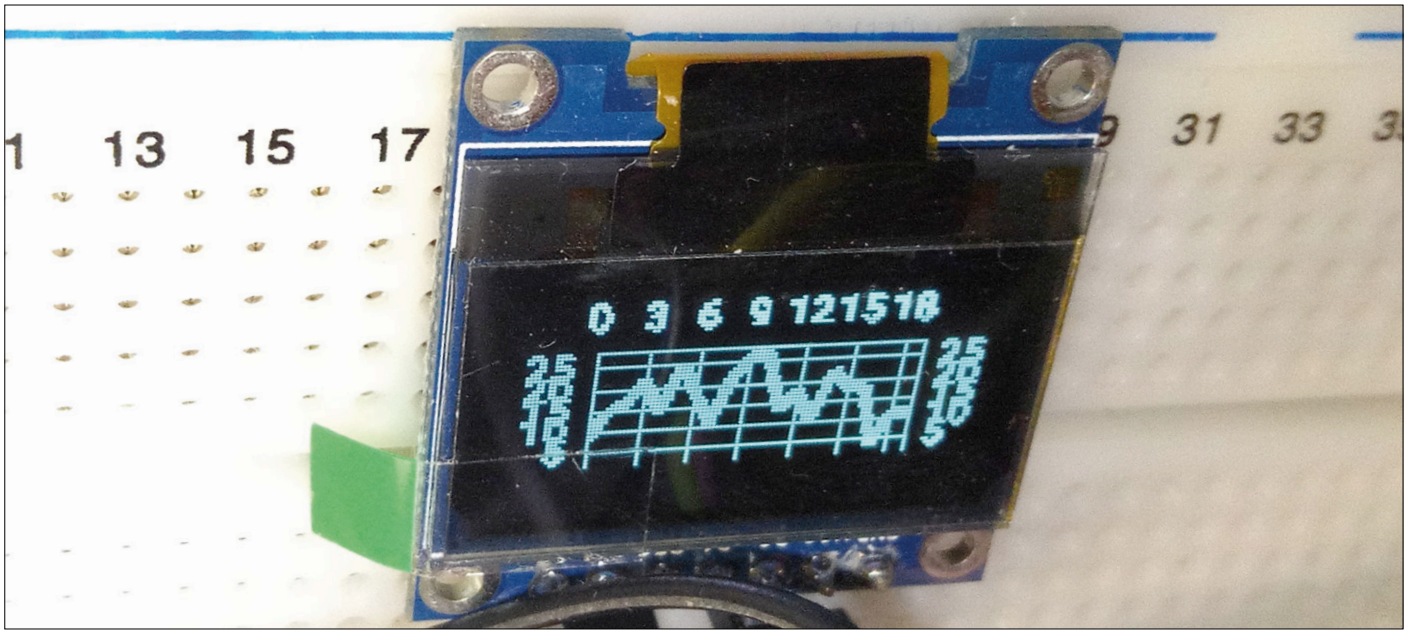


Bild 5. Dank Android mit überschaubarem Aufwand realisierbar - ein attraktives Liniendiagramm.

In unserem Fall sind hierzu drei Schleifen erforderlich:

```
byte counter=0;
byte store=0;
int counter2=0;
for(int ctr=0;ctr<8;ctr++)
    for(int x=0;x<returnedBitmap.getWidth();x++)
    {
        for (int y = ctr*8; y < (ctr+1)*8; y++)
        {
```

Als Nächstes haben wir die undankbare Aufgabe, je acht Pixel in ein Byte zusammenzufassen. Dazu beschaffen wir im ersten Schritt die Farbinformationen des jeweiligen Pixels, um es danach durch einen einfachen Vergleich der Farbkomponenten in „Schwarz“ oder „Weiß“ umzuwandeln:

```
int myVal=returnedBitmap.getPixel(x,y);
if(Color.red(myVal)!=255 || Color.
   green(myVal)!=255 || Color.blue(myVal)!=255)
        { //Set
       store=(byte)(store | ((byte)(1 <<
            (counter))));
        }
```

Das eigentliche Herausschreiben erfolgt in Achterschritten. `store` wird dabei unter Nutzung von Bitoperatoren schrittweise mit Informationen befüllt – sobald ein Wort „fertig“ ist, wandert es in `myDisplayBuffer`.

```
            counter++;
            if(counter==8) {
                myDisplayBuffer[counter2]=store;
                store=0;
                counter=0;
                //Byte raus
                counter2++;
            }
        }
    }
```

Zu guter Letzt schicken wir einige Kommandos in Richtung des Displaycontrollers, um einen definierten Zustand herzustellen. `sendData` jagt die Informationen sodann über den SPI-Bus zum Display:

```
    SendCommand(CMD_RESETCOLADDR);
    SendCommand(CMD_RESETPAGEADDR);

    sendData();

}
```

Der Lohn der Mühen ist das in **Bild 5** gezeigte Diagramm.

Fazit

Wir sind am Ende unserer Experimente mit Android Things angelangt: Es ist eine Plattform, die zwar nicht für jedermann geeignet ist, aber all jene, die mit ihr etwas anfangen können, um eine immense Fähigkeitsvielfalt bereichert. Viele Aufgaben lassen sich mit klassischen Achtbittern nicht oder nur extrem schwer realisieren - wer einen leistungsfähigen 32-bit-Prozessrechner dazuholt, spart sich Zeit und Arbeit. In diesem Sinne wünschen wir Ihnen viel Erfolg! ⏮

(160369)

Weblinks

[1] www.elektormagazine.de/160361

[2] www.elektormagazine.de/150520

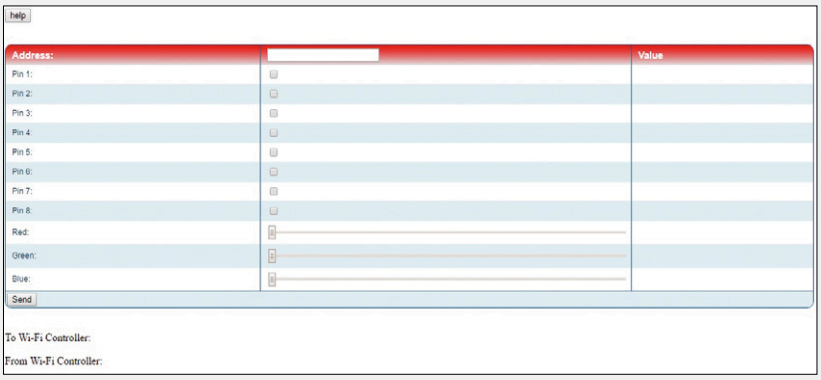
[3] www.elektormagazine.de/160369

Projekt 2.0

Korrekturen, Updates und Leserbriefe

Die Rückkehr des WLAN-Controller-Boards

Elektor 11/2016, S.22 (150402)



Die Steuerung per PC (über die Webseite 150402.html) war bisher mit den Browsern IE und Edge nicht möglich. Wenn Webseiten lokal auf der Festplatte gespeichert sind, erlauben diese Browser nicht, Daten lokal zu speichern (in diesem Fall die einmal eingegebene IP-Adresse des WLAN-Controller-Boards). Clemens Valens aus dem Elektor-Labor hat daher eine neue Version der Webseite erstellt; sie fragt den Typ des Browsers ab und unterbindet die lokale Speicherung, wenn diese nicht möglich ist. Die Daten aus dem Formular werden aber trotzdem gesendet, womit die Steuerung funktioniert. Die neue Version der Webseite findet man auf der Labs-Seite www.elektormagazine.com/labs/remake-wifi-controller-board-150402 sowie auch auf der Projektseite www.elektormagazine.de/150402.

D-Watt

Elektor 12/2016, S.42 (150115)

Im Artikel heißt es auf S. 44 rechts unten: „Die Verstärkung ist 1, mit einem Widerstand statt der Brücke für R3 kann sie jedoch etwas angehoben werden (A = R4/R3 + 1)“.
Richtig ist: „Die Verstärkung ist 1, R3 bleibt hierbei unbestückt; durch Bestücken von R3 kann die Verstärkung angehoben werden (A = R4/R3+1)“.

D-Watt

Elektor 12/2016, S.42 (150115)

Im Artikel wurde nicht erwähnt, wie P1 eingestellt wird. Es geht folgendermaßen: Nachdem alles verbunden ist, schalten Sie den Verstärker ein und prüfen die PWM-Frequenz beider Kanäle, sie sollte 400 kHz betragen. Messen Sie die Frequenz an L1 (Verbindung zu T1) ohne Eingangssignal und justieren Sie P1, um die Frequenz anzupassen. Falls Sie über kein geeignetes Messinstrument zum Messen der Frequenz verfügen, wird Poti P1 für jeden Kanal in Mittelstellung gebracht. Sie können das auch auf Elektor /Labs im Update nachlesen (www.elektormagazine.com/labs/200w-class-d-audio-power-amplifier-150511).

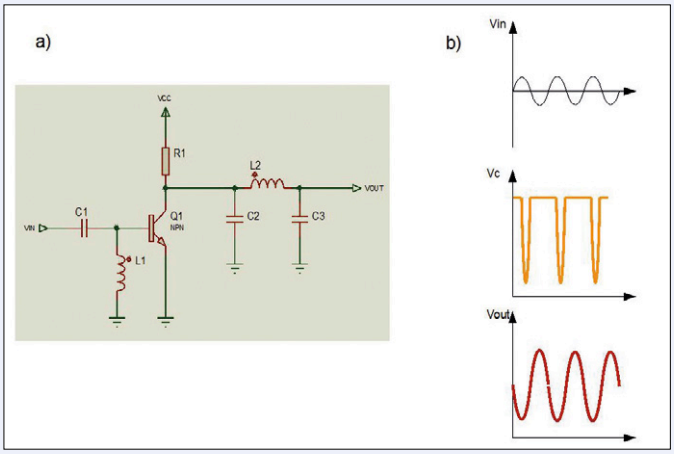
Klasse A,B,C,D,E,F,G,H: Wo ist der Unterschied?

Elektor 11/2016, S.10 (150527)

Im Artikel heißt es, dass ein Klasse-B-Verstärker theoretisch bis 100 % Wirkungsgrad haben könne. Das deckt sich nicht mit meinen Kentnissen und den Angaben in der Literatur. Dort findet man (für sinusförmige Signale) einen maximalen Wirkungsgrad von pi/4 = 78,53... Prozent. Es wird ebenfalls gesagt, dass sich mit MOSFETs ein höherer Wirkungsgrad erreichen lasse als mit bipolaren Transistoren. Ich wüsste nicht, was dafür die Begründung sein könnte, da beide Transistoren im ungesättigten Betrieb arbeiten.
Weiter heißt es, dass ein Klasse-C-Verstärker maximal 70 % Wirkungsgrad erreichen kann. Das sehe ich anders. Theoretisch kann man mit Class C durch einen kleiner werdenden Stromflusswinkel beliebig nahe an 100 % kommen. In der Praxis werden bis zu 90 % erreicht, man muss aber dann mit hohen Strömen arbeiten. Ordentliche Wirkungsgrade lassen sich nur erreichen, wenn man den Kollektorwiderstand R1 aus Bild 3 durch eine Induktivität ersetzt.

Weiter wird gesagt, dass man für Klasse-E-Stufen sehr schnell schaltende Transistoren braucht, wie bei Class D. Auch das deckt sich nicht mit meinen Kenntnissen. Man kann Schalter verwenden, deren Schaltzeiten im Bereich der Betriebsperiode/2 liegen. Oder umgekehrt: Man kann mit schnellen Transistoren Signale mit sehr hohen Frequenzen verstärken. Ein entscheidender Vorteil von Class E wird leider nicht genannt, nämlich, dass die Ausgangskapazität des Schalters nicht nachteilig wirkt.

Martin Oßmann



Hallo Martin,
Sie haben völlig Recht bezüglich Class B; das ist ein Fehler im Artikel. Der maximale Wirkungsgrad beträgt 78 % (wie Sie richtig festgestellt haben), zumindest für Sinussignale. Allerdings denke ich, dass MOSFETs in den üblichen Designs schon einen leicht besseren Wirkungsgrad als Bipolartransistoren erreichen. Und das hauptsächlich, weil sie es erlauben, näher an die Betriebsspannung für die Endstufe zu kommen, wodurch Verluste vermieden werden.
Zu Class C: Sie haben theoretisch auch hier Recht, die Effizienz kann bis zu 100 % betragen, wenn Transistoren und passive Komponenten ideale Bauteile wären. Die 70 %, die ich hier angegeben habe, entstammen eigenen Erfahrungen mit tatsächlichen HF-Verstärkern. Es ist eine echte Herausforderung, bei UHF-Frequenzen (und darüber) einen besseren Wirkungsgrad zu erzielen. Aber ich hätte das besser präzisieren sollen.
Zu Class E: Hier denke ich nicht, dass ein so langsamer Transistor in einer Klasse-E-Anwendung funktionieren könnte (aber ich bin hier kein Experte). Siehe zum Beispiel hier: http://people.physics.anu.edu.au/~dxt103/160m/class_E_amp_design.pdf.

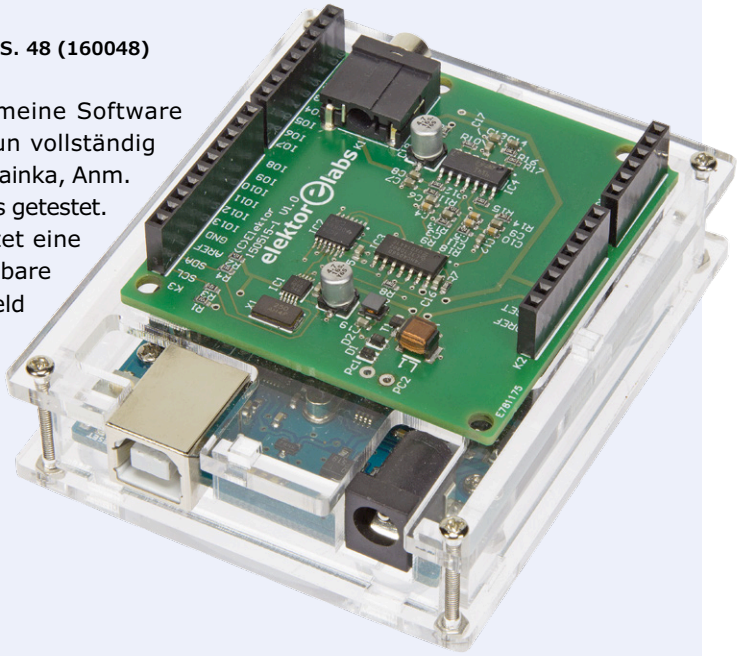
Robert Lacoste

Elektor SDR Reloaded

Elektor 07-08/2016, S. 80 (150515), Elektor 09/2016, S. 48 (160048)

Hallo Jan,
ich freue mich sehr, euch mitzuteilen, dass meine Software G8JCFSDR (Build 274) das Elektor SDR-Shield nun vollständig unterstützt. Ich war schon in Kontakt mit Burkhard [Kainka, Anm. der Red.], er hat die „Reloaded"-Version von G8JCFSDR bereits getestet.
Die Elektor-SDR-Shield-Unterstützung von G8JCFSDR bietet eine 1-Hz-Abstimmpräzision ohne Glitching, eine leicht anwendbare Kalibriereinrichtung (um sicherzustellen, dass euer SDR-Shield immer punktgenau auf der Frequenz liegt) sowie eine eingebaute Arduino-Firmware-Upload–Funktion, um das Upgrading einfach zu machen.
Das Programm G8JCFSDR ist zum Download erhältlich unter www.g8jcfsdr.dyndns.org.
Ich würde mich sehr freuen, wenn ihr eure Leser über diese neue Möglichkeit informieren könntet.

Peter, GM8JCF
g8jcf@g8jcfsdr.dyndns.org



Hallo Peter,
ich freue mich über deine Nachricht und darüber, das Update an unsere Leser weiterzuleiten. Wir werden auch deine E-Mail-Adresse zusammen mit dem Update veröffentlichen.

Jan Buiting, PE1CSI

PC-Watchdog

Elektor 07-08/2016, S.97 (140344)

Auf der Projektseite (www.elektormagazine.de/140344) steht eine neue Projekt-Software zum Download bereit.

Alles Gute zum 40., PET!

Nichts löten, und BASIC für jedermann

Zur Consumer Electronics Show (CES) im tiefgefrorenen Chicago des Januars 1977 gab es eine ganz besondere Premiere. Dem staunenden Publikum wurde der weltweit erste Personal Computer mit einer kompletten, betriebsbereiten Ausstattung vorgestellt: Der Commodore Personal Electronic Transactor, oder viel kürzer: der PET 2001.

Von **Karl-Ludwig Butte**
(Deutschland)

Foto: Wikimedia Commons

Spätestens, seit zwei Jahre zuvor die Zeitschrift „Popular Electronics“ mit der Veröffentlichung eines Artikels über den Altair 8800 für Furore sorgte und diesen als Bausatz anbot, brodelte es in der damaligen Microcomputer-Enthusiasten-Szene (**Bild 1**). Überall wurde fleißig an Eigen-Kreationen gewerkelt, erschienen unzählige Bücher über „How to build your own working computer“ (**Bild 2**) und schossen Computer-Clubs, wie z.B. der Homebrew Computer Club [1], wie Pilze aus dem Boden. Doch jetzt war die Zeit reif für einen Computer, der schon alles mitbrachte, was man für den Einstieg in das neue Computerzeitalter brauchte: CPU, Stromversorgung, Hauptspeicher, Bildschirm, Tastatur und Massenspeicher – alles in einem Gehäuse! Und der Clou: Wenige Sekunden nach dem Einschalten meldete sich der Commodore BASIC Interpreter, bereit jeden BASIC-Befehlswunsch des Anwenders sofort auszuführen. Kein Einstellen von Binärschaltern mehr, wie beim Altair 8800, und kein Eintasten von Hexwerten, wie beim KIM-1 (der erste Versuch von Commodore, auf diesem Zukunftsmarkt Fuß zu fassen). Nein, klare verständliche (BASIC-)Worte, wie z.B. `GOTO`, `PRINT` oder `IF …THEN`, genügten, um dem PET 2001 die unglaublichsten Fähigkeiten einzuprogrammieren – eine echte Revolution, deren Ausmaß wir uns heute, angesichts von 8-Kern-CPUs, Gigabyte-Speichern, Terabyte-Festplatten und

ESTD 2004
www.elektor.tv

Retronik ist eine Rubrik, die antiker Elektronik und legendären Elektor-Schaltungen ihre Reverenz erweist. Beiträge, Vorschläge und Anfragen telegrafieren Sie bitte an Jan Buiting (editor@elektor.com).

Graphical-User-Interfaces, kaum mehr vorstellen können.

Die Hardware

Was bekam man nun für 595 $ (das entspricht 2350 $ nach heutigem Wert [2])? Das Herz des PETs war eine 6502 CPU, die auf 4 KB Hauptspeicher zurückgreifen konnte. Satte 8 KB Hauptspeicher gab es gegen einen Aufpreis von 200 $ [3]. Für die Ausgabe war ein 9“ großer, monochromer Bildschirm mit weißer oder grüner Schrift eingebaut. Die Tastatur bestand aus 73 Tasten, die Commodore aus seiner Tischrechenmaschinen-Fertigung übernommen hatte. Das hatte den Vorteil, dass die Tastatur nicht so viel Platz beanspruchte wie eine echte Schreibmaschinen-Tastatur, und daher noch ein Kassettenrekorder für handelsübliche Compact-Kassetten links neben ihr untergebracht werden konnte. Darauf ließen sich Daten und Programme zur späteren Verwendung speichern. Ein zweiter Rekorder ließ sich extern über den sogenannten Datasette-Port anschließen. Zwei weitere Schnittstellen sicherten dem PET eine großartige Zukunft. Da war zunächst der sogenannte Userport. An ihm waren unter anderem die digitalen I/O-Anschlüsse des 6502-Prozessors zugänglich, an die man eigene Hardware anschließen konnte. Die meisten Personal-Computer dieser Zeit, wie z.B. der Apple II, TRS-80 oder der C64, hatten solch eine Erweiterungsmöglichkeit. Danach verschwand diese Option aus nachfolgenden Generationen von PCs, bis sie unter dem Namen „GPIO-Pins“ 2012 beim Raspberry Pi eine Renaissance erlebte. Mit dem Userport kam man nicht nur den Hobby-Bastlern, sondern auch den überall neu gegründeten Unternehmen entgegen, die Zusatzhardware für die Personal-Computer entwickelten. Die zweite Schnittstelle, ein IEEE488-Bus, wurde von Commodore selbst genutzt, um Peripheriegeräte anzuschließen, wie z.B. Drucker und Diskettenstationen (**Bild 3**). Die Verwendung dieses Bussystems war in mehrfacher Hinsicht bemerkenswert. Die Peripheriegeräte wurden in einer Daisychain hintereinander an den PET angeschlossen, konnten aber trotzdem parallel durch diesen angesteuert werden. Commodore musste sich anfangs nicht festlegen, wieviel und welche Peripheriegeräte später dazukommen sollten. Der Bus bot mit maximal 15 Geräten mehr als genug Möglichkeiten. Außer dieser einen IEEE488-Schnittstelle musste keine weitere Hardware auf dem Mainboard des PET verbaut werden. Das bedingte zwar, dass die Peripheriegeräte ihre eigene Intelligenz in Form eines vollständigen Mikrocontrollersystems mitbringen mussten, dafür ließen sie sich dann aber teurer verkaufen. Zusätzlich sparte Commodore bei der Entwicklung des PETs wertvolle Zeit, die für einen früheren Markteintritt genutzt werden konnte. Apple ging den entgegengesetzten Weg. Für Erweiterungen standen sieben freie Steckplätze auf dem Motherboard zur Verfügung. Darin konnte z.B. ein Diskettencontroller eingesteckt werden, der lediglich das elektrische Interface zur Laufwerks-Elektronik darstellte. Die gesamte Steuerung der maximal zwei anschließbaren Floppy-Laufwerke musste die Apple II CPU selbst übernehmen. Der PET dagegen übertrug lediglich ein Kommando an die Floppy-Station und konnte sich dann, z.B. bei dem Befehl „Kopiere Diskette 1 auf Diskette 2“ wieder anderen Aufgaben widmen (während die Diskettenstation selbsttätig den Befehl ausführte).

Die gesamte Hardware war in einem massiven Stahlblechgehäuse untergebracht, welches sich wie die Motorhaube eines Autos öffnen ließ. Sogar ein Metallstab zum Abstützen des Deckels war vorhanden (**Bild 4**). Diese stabile Konstruktion trug sicherlich auch mit dazu bei, dass etliche PETs bis heute überlebt haben. Ab und zu werden sie heute zu Preisen zwischen 200 und 800 € auf den bekannten Auktionsseiten angeboten. Oft kommen die Geräte in einem bedauernswerten Zustand bei ihren neuen Besitzern an. Wenn man Glück hat, sind sie nur stark verschmutzt (**Bild 5**) und strahlen nach einer entsprechend ausgedehnten Putz-Session wieder in altem Glanz (**Bild 6**).

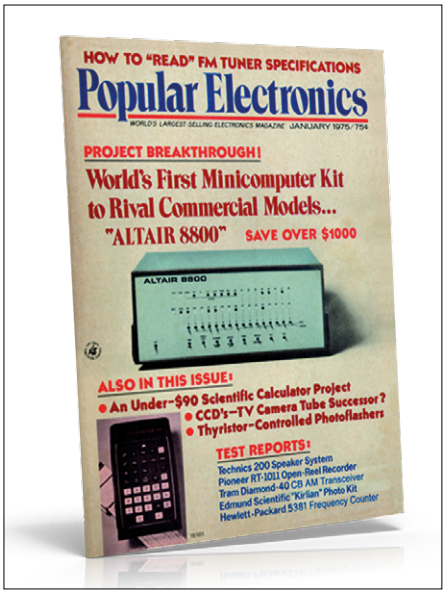


Bild 1. Cover der „Popular Electronics“ vom Januar 1975 (Ziff Davis Publishing Company).

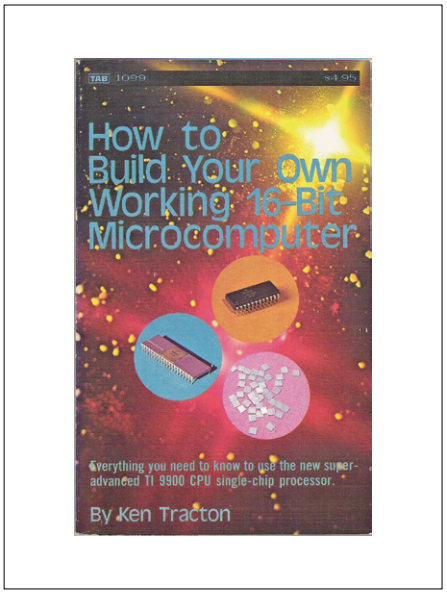


Bild 2. Eins von zahlreichen Selbstbau-Büchern.

Die Software

Der BASIC-Interpreter des PET war Betriebssystem und Programmier-Entwicklungsumgebung in einem und befand

Bild 3. Eine voll ausgebaute Anlage mit PET 2001, Drucker und Diskettenstation.

Bild 4. Das Gehäuse ließ sich durch das Lösen von nur zwei Schrauben öffnen.

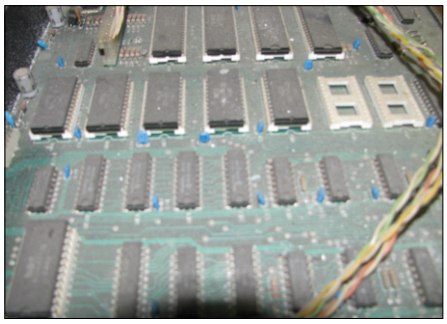


Bild 5. Zustand des Mainboards bei der Ankunft beim Autor.

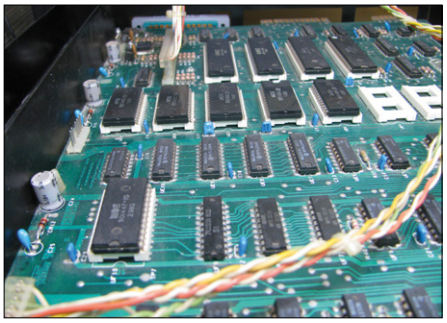


Bild 6. Zustand des Mainboards nach der Putz-Session.

Bild 7. BASIC-Prompt nach dem Einschalten (Tom Knox; gemeinfrei; Wikipedia).

sich in 14 KB ROM (**Bild 7**). Er beruhte auf Microsoft BASIC und war durch einen sogenannten „Full-Screen-Editor" erweitert worden [4]. Damit konnte man Programmzeilen, die auf dem Bildschirm sichtbar waren, nachträglich ändern, indem man den Cursor auf die entsprechende Zeile bewegte und sie dann editierte. Nach Drücken der Return-Taste wurden die Änderungen übernommen. Für damalige Verhältnisse ein unglaublicher Komfort! Neben den üblichen BASIC-Befehlen gab es auch noch die Befehle `PEEK` und `POKE`, mit denen sich einzelne Bytes gezielt aus dem Speicher lesen bzw. in den Speicher schreiben ließen.
Variablennamen wurden nur anhand der ersten beiden Zeichen unterschieden. Sie durften zwar länger sein, aber die zusätzlichen Zeichen wurden ignoriert. Für den BASIC-Interpreter waren also z.B. die Variablen `MAUS` und `MANTEL` dieselbe Variable „`MA`".

Commodore-BASIC kannte drei verschiedene Datentypen: 40-Bit-Gleitkommazahlen, 16-Bit-Integer-Zahlen und Zeichenketten (Strings) mit bis zu 255 Zeichen. Diese wurden anhand eines speziellen Zeichens am Ende des Variablennamens erkannt. Zeichenketten erhielten ein `$`-Zeichen, während Integer-Variablen an einem `%`-Zeichen erkannt wurden. Fehlte ein solches Erkennungszeichen, wurde die Variable als Gleitkommazahl behandelt. Variablen konnten nicht deklariert werden. Wurde eine neue Variable erkannt, reservierte der Interpreter eine dem Datentyp entsprechende Anzahl Bytes im Speicher. Das hatte bei Tippfehlern verheerende Folgen und man wunderte sich über das völlig unerklärliche Verhalten seines Programms. Jeder, der schon einmal eine Variable `RCHNUNG$` eingetippt hat, aber `RECHNUNG$` meinte, weiß ein Lied davon zu singen. Oft half es, das Programm mit einem unbeteiligten Dritten durchzugehen. Dieser sah erfahrungsgemäß solche Tippfehler viel eher als man selbst.
Eine BASIC-Anweisung konnte zur sofortigen Ausführung einfach eingegeben werden. `PRINT 2*SQR(25)` <Return> brachte augenblicklich das Ergebnis „10" auf den Bildschirm. Wurde eine Zeilennummer vor die Anweisung geschrieben, wartete der PET mit der Ausführung, bis man den Befehl `RUN` eingab. Spätestens nach zwei bis drei Programmierversuchen hatte der Benutzer gelernt, dass es sinnvoll ist, diese Zeilennummern mindestens in Zehnerschritten zu vergeben. Hatte man nämlich ein zehnzeiliges Programm geschrieben und die Zeilennummern 1, 2, 3, … 10 vergeben, war es nicht möglich, zwischen die Zeilen 4 und 5 eine elfte Zeile einzufügen, ohne die Zeilen 5 bis 10 manuell in die Zeilennummern 6 bis 11 abändern zu müssen. Dabei durfte man auch nicht vergessen, die Ziele von bedingten (`IF`) und unbedingten (`GOTO`) Sprungbefehlen entsprechend anzupassen. Einen `RENUMBER`-Befehl, der all dies automatisch erledigte, gab es anfangs nicht. Doch die engagierte Programmierergemeinschaft (heute nennt man das „Community") sann auf Abhilfe. Da schnell klar war, dass ein BASIC-Programm schlecht ein anderes BASIC-Programm neu durchnummerieren konnte, mussten schwerere Geschütze aufgefahren werden. Die Lösung hieß Assembler – ein in Maschinensprache geschriebenes und in einem geschützten Speicherbereich untergebrachtes Programm sollte in der Lage sein, dieses Wunder zu vollbringen. Zum Glück brachte der PET den sogenannten *Machine Language Monitor* mit (anfangs nur auf Kassette, später dann im ROM). Jetzt wurde es erst richtig interessant und jeder PET-Programmierer, der etwas auf sich hielt, kannte den „Geheimbefehl", um in die Tiefen der Assembler-Programmierung vorzustoßen: `SYS 1024` <Return>. Und tatsächlich: Maschinensprache half in vielen Fällen, wo man mit BASIC nicht weiterkam. Fachzeitschriften, Bücher und gedruckte Newsletter von zahlreichen Computerclubs druckten seitenweise Hexcode-Listings ab. Damit konnte man essentielle Dinge implementieren, wie z.B. den `RENUMBER`-Befehl. Auch praktische Dinge, wie z.B. den Anschluss eines Barcodelesers, mit dem man andere Programme als Strichcode einlesen konnte, ließen sich realisieren.

Die Anwendungen

Spiele waren sicherlich ein Hauptanwendungsbereich für den PET 2001. Dabei dürfen Sie sich natürlich keine bunten Mega-Pixel-Grafik-Spiele vorstellen, wie sie heute auf jedem Smartphone laufen. Der PET hatte nicht einmal eine echte Grafik-Anzeige, bei der man jedes Pixel einzeln ansteuern konnte. Nein, es gab lediglich eine Reihe von Grafik-Symbolen mit Linien, Blöcken oder Symbolen, die in das Raster der auf dem Bildschirm darstellbaren Buchstaben und Zahlen eingepasst waren. „Blockgrafik" nannte

Weblinks

[1] http://de.wikipedia.org/wiki/Homebrew_Computer_Club
[2] www.minneapolisfed.org
[3] www.kuto.de/cbmmuseum/cbm_pet.html
[4] http://de.wikipedia.org/wiki/Commodore_BASIC

man das und war stolz darauf, damit Balken-Diagramme, Bildschirmmasken und eben Spiele anzeigen zu können.
Mathematische Anwendungen reichten von einfachen Berechnungen über Tilgungspläne bis zu ausgefuchsten Lineare-Gleichungslöser-Programmen. Dabei halfen die in BASIC verfügbaren mathematischen Befehle, wie z.B. `LOG` (Logarithmus) und `SQR` (Quadratwurzel).

Mit Erscheinen von Druckern und Diskettenstationen wurden dann auch kommerzielle Anwendungen, wie z.B. Auftragsbearbeitung, Rechnungsschreibung und Lagerverwaltung möglich. Gerade Kleinbetriebe wagten den Schritt, weil eine Commodore-Anlage mit Computer, Drucker und Diskettenstation immer noch um Klassen preiswerter war, als die Einsteigermodelle bei den etablierten Groß-EDV-Herstellern. Eine wirkliche Killer-App, wie wir heute sagen würden, kam 1979 in der Gestalt des ersten Tabellenkalkulationsprogramms Visicalc auf den Markt. Ursprünglich für den Apple II geschrieben, wurde die Software kurze Zeit später auch auf den PET portiert. Auch die ersten Text-Editor-Programme brachten den PET in manches Büro.
Viele Hobby-Programmierer sammelten ihre ersten Erfahrungen mit dem Anschluss von zusätzlicher Hardware am Userport. So konnte man immer wieder Anleitungen finden, wie man dem PET Töne entlocken konnte. Auch die eine oder andere kleinere Mess- und Steuerungsaufgabe ließ sich mit dem PET lösen.

Fazit

Der PET hat nicht nur Computer-Geschichte geschrieben, sondern auch Wirtschaftshistorie. Mit ihm wurde eine Entwicklung eingeleitet, die die IT an die Arbeitsplätze in den Unternehmen brachte und einen völlig neuen Markt entstehen ließ. Ganze Berufsgruppen, wie z.B. die Setzer in den Druckereien, verschwanden, und neue, wie z.B. Programmierer und IT-Berater, etablierten sich innerhalb kürzester Zeit.

Das ganze Ausmaß der Veränderungen haben wohl nur wenige erfasst, die Ende der 70er Jahre des vorigen Jahrhunderts ihre ersten Gehversuche in der neuen Computerwelt mit dem PET unternahmen. ⏮

(160353)

Delayed Trigger

Zusammengestellt von Jan Buiting

Seit dem Start der Retronik-Serie im Jahre 2004 generierten diese Elektor-Artikel umfangreiche Reaktionen unserer Leser, die sich nicht nur auf schriftliche Rückmeldungen beschränkten. Nachfolgend sehen Sie ein paar Beispiele von Sachspenden und anderen Beiträgen, die über die Jahre bei uns einliefen, sowie Hinweise auf den ursprünglichen Artikel, der als „Trigger-Quelle" diente. Aus Datenschutzgründen werden die Namen der Personen nicht genannt, wenn sie das nicht ausdrücklich erlaubt haben.

Objekt 04: Elektor MOS-Uhr 5314

Der Artikel zur MOS-Uhr triggerte ganze Reaktionsfluten. Alan Woodman beglückt uns mit seiner hier abgebildeten Scope-Clock. Reginald Neale packte das größere IC MM5316 auf eine Lochrasterplatine. Bernd Rinke scannte den 1974er Original-Artikel, überarbeitete die Platinen-Layouts – all das steht jetzt kostenlos unter www.elektormagazine.de/160102 zum Download zur Verfügung. Philippe Decock schickte eine Ladung lustiger Anekdoten seiner denkwürdigen Erinnerungen und verwies auf einen Missing-Link!
Status: Wir danken und schmunzeln.
Trigger 04: MOS-Uhr 5314 (1974), Elektor 9/2016.

Objekt 05: Roberts Radio Typ 49041

Die Welt ist klein! Sowohl mein Hausarzt als auch mein Zahnarzt lesen Elektor. Sie sind immer wieder für einen Spruch zum Inhalt gut und lassen sich gerne über die zugrundeliegende Physik aus. Zusammen mit dem Rezept überreichte mir mein Hausarzt Gaspard Knops auch noch Roberts 1950er Radio, das mit Batterieröhren bestückt war. Sein Kommentar war, dass er meinen Beitrag über das Colette-Radio von Philips sehr genossen und auch die Story über Ronald Dekkers P87BQ-Radio von Pye toll gefunden hätte. Ein Grund für die Beliebtheit dieser Batterieröhren bei den Elektor-Lesern könnte darin liegen, dass sie schon mit relativ niedrigen Anodenspannungen laufen.
Status: Muss überholt werden.
Trigger 05: Das Kofferradio P87BQ von Pye, Elektor Mai 2015.

Objekt 06: pH-Meter von Electrofact

Viele Elektroniker haben sich schon mit Medizin-Elektronik beschäftigt. Ich vermute, dass dieses voluminöse pH-Meter aus dem Röhrenzeitalter am besten in der sicheren Umgebung eines Analyse-Labors betrieben wurde. Electrofact war eine niederländische Firma. Das Messgerät wurde mir zusammen mit kompletter Glasware und einigen Sensoren übergeben. Die Dokumentation hat allerdings nicht überlebt, was (abgesehen von den Röhren) den Einblick in die Funktion erschwert. Aber immerhin: elektrisch funktioniert die Kiste.
Status: Weitere Recherche & Restauration.
Trigger 06: Radiometer PHM22 /10PHA928a zur Bestimmung von pH, O_2 und CO_2 im Blut, Elektor 9/2013.

Lassen auch Sie sich post-triggern, um hier aufgeführt zu werden!
E-Mail: **editor@elektor.com**; Betreff: **Trigger [nn]**.

(Fast) alles was Sie schon immer wissen wollten über ...

Smarte Gebäude

Smarte Gebäudetechnik ist momentan in aller Munde. Wir baten Emmanuel Francois um einige Erläuterungen zu diesem Hype.

Q *Gibt es eigentlich einen großen Unterschied zwischen smarten Gebäuden und der Heimautomatisierung?*

A Ich sehe da zwei größere Unterschiede: Beim smarten Heim gibt es nicht so viel Technik wie bei einem smarten Gebäude. Außerdem sind Einsparungen bei einem smarten Gebäude viel wichtiger.

Noch ein Unterschied: Bei einem smarten Gebäude liegt der Fokus mehr auf lokaler IT-Infrastruktur. Das BMS (**B**uilding **M**anagement **S**ystem) ist eher „analog" und nicht notwendigerweise interaktiv oder fernsteuerbar. Bei einem smarten Heim hingegen ist alles „voll digital" und über das Internet fernsteuerbar. Mit den neuerdings für iOS und Android erschienenen Home-Apps für Mobilgeräte kann man das eigene Heim bzw. seine Geräte überwachen und fernsteuern – ein gutes Beispiel für diese Unterschiede.

Q *Gibt es da ähnliche (Software-)Protokolle wie bei der Heimautomatisierung?*

A Zunächst bedeutet „smart" beim smarten Heim wie bei smarten Gebäuden den Einbezug von IT-Infrastruktur. Daher braucht es in beiden Feldern eine IP-Verkabelung. Hier setzt sich TCP/IP aufgrund der Internet-Kompatibilität immer mehr durch.

Das Netz muss aber auch flexibel und skalierbar sein. Naheliegend sind hier Funklösungen. Hierfür gibt es bereits bewährte Protokolle wie WLAN oder Bluetooth (LE), weshalb diese auch eingesetzt werden. Zum Beispiel setzen schon viele moderne LED-Beleuchtungssysteme auf Bluetooth LE. Diese Systeme sind dann an das IP-Netzwerk angebunden.

Bei lokalen, drahtlosen Netzwerken ist ZigBee, EnOcean [1] und Thread [2] am weitesten verbreitet. KNX ist zwar ebenfalls ein bekanntes Protokoll, wird aber in Funk-Netzwerken kaum eingesetzt. Es gibt natürlich noch andere Protokolle, doch für gewöhnlich fällt die Entscheidung zugunsten offener Protokolle. Beim smarten Heim und smarten Gebäuden findet man kaum ein proprietäres Protokoll.

Q *Was sind die Vorteile von smarten Gebäude-Systemen?*

A An erster Stelle: Effektivität! Das gilt sowohl für den Energieverbrauch als auch für die Raumausnutzung. Zum Beispiel bleiben in Frankreich fast die Hälfte der in Bürogebäuden verfügbaren Schreibtische aufgrund von Tele-Arbeit ungenutzt. Wenn nur etwa 10 % Arbeitsplatzfläche in einem Bürogebäude mit 10.000 m² wegfielen, könnte das bis zu 1.000.000 € pro Jahr einsparen. Deshalb müssen Büroflächen flexibler nutzbar werden.

Zweitens kann man durch eine gut angepasste Arbeitsplatzumgebung die Produktivität der Mitarbeiter steigern. Dies bezieht sich auf Komfort, saubere Luft und gute Beleuchtung. Und drittens gibt es noch den Punkt Gebäude-Management. Mit einem smarten BMS kann man die Verwaltung von Gebäuden optimieren.

Q *Welches Projekt ist Ihrer Ansicht nach ein besonders gutes Beispiel dafür, was heute mit smarter Gebäude-Technik möglich ist?*

A Mein Favorit ist „The Edge" in Amsterdam [3]. Es ist in puncto Raum und Energie besonders effektiv. Anfänglich wurde noch eine Fläche 100.000 m² projektiert. Doch dank smarter Raum-Lösungen waren dann 60.000 m² ausreichend, was fast die Hälfte der geplanten Fläche erübrigte und so enorme Kostensenkungen möglich machte.

Die französische Firma Bouygues baut gerade 10.000 voll vernetzte „smart homes" auf der Basis von Flexom [4] – pro Jahr. Dank Funkvernetzung und dem EnOcean-Protokoll schaffen sie das fast ohne zusätzliche Kosten. Dies spart nämlich viel Geld bei der Hausinstallation. Darüber hinaus wird NFC für die Inbe-

triebnahme der Systeme eingesetzt, was einen Zeitbedarf von nur rund 20 Minuten pro Haus erfordert, um dessen smarte Technik in Betrieb zu nehmen.

Q *Wann wird sich diese Technologie durchsetzen?*

A Dies hängt stark vom Vertrauen der Anwender in die Technik ab. Aus diesem Grund haben wir an einem Standard für smarte Gebäude mit der Bezeichnung Ready2Services gearbeitet, der in Standards wie HQE [5] und BREEAM [6] integriert werden kann. Da wir in einem sich schnell ändernden IT-Umfeld leben, geben diese Standards den Investoren die Sicherheit, dass ein smartes BMS aktuell bleibt und zukunftssicher ist.

Q *Integration zwischen smartem Heim und smartem Auto, wann wird das passieren?*

A Vermutlich in zehn Jahren werden kaum noch Autos mit Verbrennungsmotoren verkauft werden, und fünf Jahre später wird man kaum noch welche auf der Straße sehen. Ich bin mir sicher, dass Länder wie Schweden schon 2025 Einschränkungen für den Betrieb von Autos mit Verbrennungsmotoren beschließen. Als Resultat wird dann quasi jedes Auto elektrisch fahren. Und dann wird nicht nur der Akku, sondern das komplette elektrische System eines Autos mit dem Energie-System von smarten Gebäuden verbunden sein.

Q *Welche Aspekte bestimmen die aktuelle Forschung?*

A Sowohl bei smarten Autos als auch beim smarten Heim geht es um den gleichen zentralen Aspekt: die Intelligenz. KI treibt hier die Innovation, denn sowohl das Auto als auch die Haustechnik müssen sich dem Anwender anpassen. Das ist eine große Herausforderung für alle Beteiligten. Sie müssen sich intensiv mit KI befassen oder aber sie landen in der zweiten Reihe. ⏮

(160259)

Wer ist Emmanuel François?

Emmanuel beschäftigt sich schon lange mit smarten Gebäuden. Im Jahre 2014 wurde er im Alter von 55 Jahren Chairman der SBA (**S**mart **B**uilding **A**lliance for Smart Cities), die er zusammen mit Serge Le Men von Newron System im Jahre 2012 gründete. Mit 140 Mitgliedern unterstützt die SBA alle Beteiligten beim Übergang von smarten Gebäuden und smarten Gemeinden in die digitale Welt. SBA hat mit Ready2Services und Ready2Grid zwei Grundlagen für smarte Gebäude und smarte Gemeinden entwickelt. Emmanuel François ist außerdem Sales Manager für Westeuropa bei der EnOcean GmbH. Im Bereich Elektrotechnik war er zuvor auch schon bei Wieland Electric France, CAPRI (Cooper Group), Soulé (ABB Group) und Merlin Gerin (Schneider Electric) tätig.

Weblinks

[1] https://de.wikipedia.org/wiki/Enocean

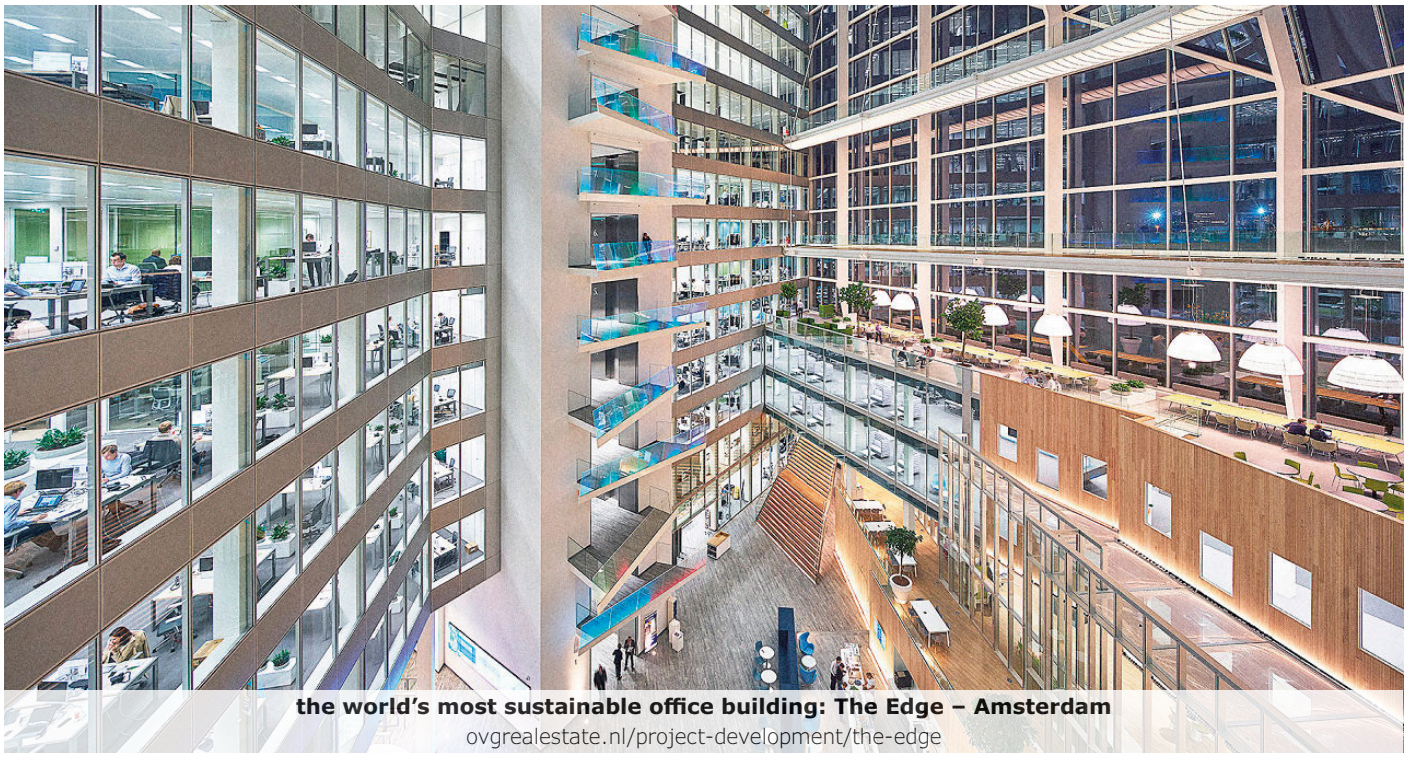
[2] https://en.wikipedia.org/wiki/Thread_(network_protocol)

[3] http://ovgrealestate.de

[4] https://flexom.bouygues-immobilier.com (französisch)

[5] https://en.wikipedia.org/wiki/Haute_Qualité_Environnementale

[6] www.breeam.com



the world's most sustainable office building: The Edge – Amsterdam
ovgrealestate.nl/project-development/the-edge

WaterDeck

Analyse des Wasserverbrauchs über das Internet

Von **Waldemir Cambiucci** (Brasilien)

Immer mehr Gegenden der Erde haben mit großem Wassermangel zu kämpfen. Die Auswirkungen auf die Bevölkerung und die Industrie sind dramatisch. Obwohl die Behörden an Strategien arbeiten, um dieses Problem zu lösen und auffordern, Wasser zu sparen, fehlen immer noch Echtzeit-Informationen über den Wasserverbrauch. Was benötigt wird, ist eine Datenerfassung über ein Web-Interface.

Das WaterDeck-Projekt stellt eine Lösung zur Überwachung und Analyse des Wasserverbrauchs in vernetzten Städten und Gebieten dar. WaterDeck sammelt Daten zahlreicher verschiedener Geräte und Sensoren. Hierzu gehören Füllstandsmesser in Behältern und Wassertanks, Durchflussmesser und Stellantriebe für Wasserpumpen. All diese Datenquellen sind mit einem Freescale-Mikrocontrollerboard verbunden. Die Messwerte der Sensoren werden gesammelt und über das Internet privaten, geschäftlichen und industriellen Stellen in Echtzeit auf einer Informationsplattform zur Verfügung gestellt. Die Informationen können auf Websites, APIs, Tablets und Smartphone-Apps eingesehen werden. Durch die Nutzung einer Cloud lassen sich auch die riesigen Datenmengen, die sich im Laufe der Zeit ansammeln, beherrschen, so dass nicht nur vergangene Entwicklungen analysiert werden können, sondern auch die Möglichkeit besteht, Berichte und Prognosen zu erstellen. Im Internet der Dinge (IoT) sind die gesammelten Daten genauso wichtig wie die Hardware aus Sensoren, Mikrocontrollern und Geräten. Die Kombination robuster Mikrocontroller mit hochqualitativen Daten schafft erhebliche Vorteile für die Nutzer. Dies gilt sowohl für private als auch für kommerzielle Anwender, die so den tatsächlichen Wasserverbrauch mit den Angaben auf der Wasserrechnung vergleichen können.

Mit einer anwachsenden Weltbevölkerung - neun Milliarden Menschen im Jahr 2050 - und dem entsprechend steigenden Wasserbedarf kann das WaterDeck-Projekt nicht nur in Krisenzeiten, sondern auch kontinuierlich ein Instrument zur Verbrauchsoptimierung und zur Überwachung sein, zu jeder Zeit und an jedem Ort auf der ganzen Welt.

Bausteine

Bei diesen Projekt haben wir uns für das Freescale-Board FRDM-K64F [1] entschieden (**Bild 1**) Das FRDM-K64F-Board von Freescale, das in Zusammenarbeit mit mbed entworfen wurde, eignet sich dank des kompakten Formats und des guten Preis/Leistungsverhältnisses des Cortex-M4-Controllers hervorragend als Basis für Prototypen aller Art. Als Entwicklungsplatine verfügt sie über Anschlüsse für Erweiterungsboards und besitzt einen integrierten USB-Flash-Programmierer. Wegen der arduino-kompatiblen Anschlüsse stellt die FRDM-K64F-Platine eine einfache, robuste und erweiterbare Plattform für Entwickler und Ingenieure dar.

Im Rahmen des „ARM Microcontroller Design Contests 2015“ (siehe **Kasten**) konnten wir eine von vier verschiedenen Projekt-Templates auswählen. Wir entschieden uns für eine Datenerfassung mit Webinterface. Über eine Middleware-Netzwerk-

Ein Gewinn mit Gewinn

Im Jahr 2015 erhielt dieses Projekt den dritten Preis im ARM Keil Microcontroller Design Contest, bei dem es darum ging, ein intelligentes Haus- oder Industriegerät auf Basis eines ARM-Cortex-M-Prozessors zu erstellen. Am Wettbewerb nahmen mehr als 700 Projekte teil.

Waldemir fasst seine Erfahrung zusammen: *Ich denke, dass der von Elektor organisierte ARM-Microcontroller-Design-Contest eine wunderbare Gelegenheit war, meine persönlichen Fähigkeiten zu verbessern und das neue FRDM-K64F-Board von Freescale kennenzulernen. Es war auch eine großartige Möglichkeit, mit anderen Enthusiasten auf der ganzen Welt in Verbindung zu treten. Und schließlich war es eine Ehre, als einer der Gewinner dieses Wettbewerbs ausgewählt zu werden. Ich bin schon ganz gespannt auf den nächsten ARM-Keil-Wettbewerb!*

komponente gelangten die eingehenden rohen Sensordaten eines Beschleunigungsmessers und eines Magnetometers auf eine Webseite. „Middleware“ bezeichnet eine Software-Ebene, die als Schnittstelle zwischen zwei ungleichen Computercodes arbeitet, beispielsweise zwischen einem Anwendungsprogramm und einem Betriebssystem. Der Benutzer kann über CGI-Scripts und JavaScript das Aufzeichnungsintervall der eingehenden Daten ändern.

Das Ziel (**Bild 2**) bestand darin, die vielfältigen Informationen der Sensoren, Wasserpegelmesser, Pumpen, Durchflussmessgeräten und so weiter zu erfassen und als Push-Meldungen über ein Web-Interface an den Endbenutzer zu senden. Auf diese Weise konnte das Dashboard auf einem beliebigen Gerät (Tablet, Smartphones oder Desktop) den in der Cloud gehosteten Datensatz anzeigen.

WaterDeck in Aktion

Für dieses Projekt haben wir eine komplette Analyseumgebung mit zwei Tanks zur Simulation des Wasserverbrauchs aufgebaut (**Bild 3**). Der erste, mit „Water Supply System“ gekennzeichnete Tank stellt die Verbindung zur öffentlichen Wasserversorgung dar. Der zweite ist der Haus-Wassertank (Domestic Water Tank), wie er in vielen Ländern üblich ist.

Der Hauswassertank ist mit drei Füllstandssensoren von ICOS [2] ausgestattet, die über normale digitale Eingänge an das Freescale-Board angeschlossen werden. Die Software erkennt, ob ein bestimmter Füllstand erreicht ist oder nicht.

In der Hauptzuleitung wird vor oder hinter der Hauptpumpe ein Durchflusssensor platziert. Damit kann ermittelt werden,

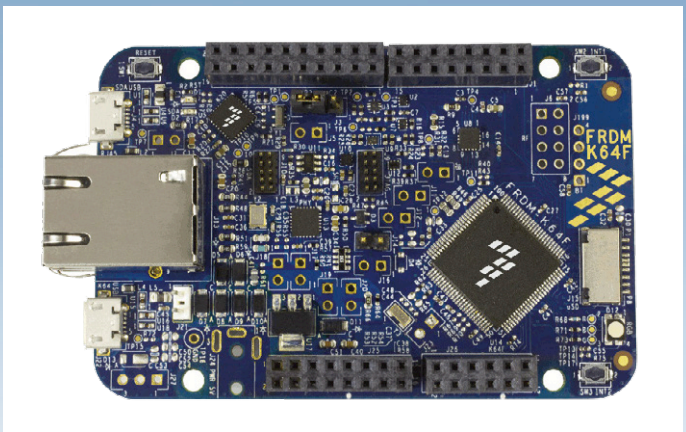


Bild 1. Das FRDM-K64F-Board von Freescale.

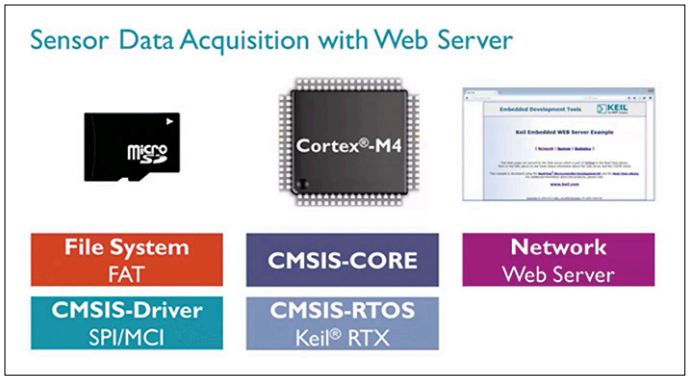


Bild 2. Akquise von Sensordaten mit Webserver.

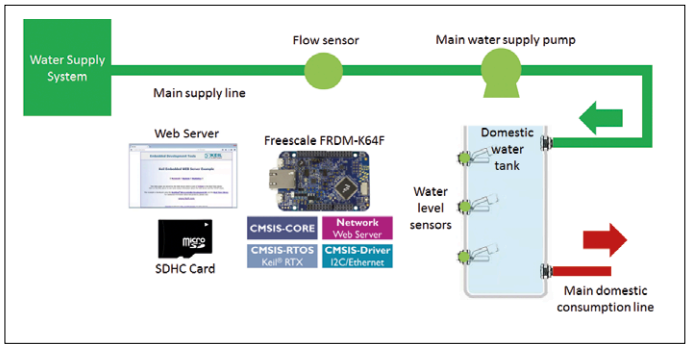


Bild 3. Mit dem FRDM-K64F-Bboard von Freescale wird der Wasserverbrauch von Privathaushalten überwacht und gesteuert. Alle Informationen werden im Internet präsentiert.

wie viele Liter pro Stunde oder Minute von der Wasserversorgung in den Hauswassertank gelangen.
Schließlich haben wir eine andere Wasserpumpe angeschlossen, um die Wasserentnahme aus dem Hauswassertank zu simulieren. Wenn diese Pumpe aktiviert wird, pumpt sie das Wasser aus dem Hauswassertank (und führt sie, damit sich unter der Versuchsanlage keine Pfütze bildet, in den primären Tank zurück).
Das FRDM-K64F-Board von Freescale ist das Herzstück dieser Simulation und steuert die digitalen Ein- und Ausgänge des Systems. Ein sehr praktisches Merkmal dieses Boards ist die Möglichkeit, alle gesammelten Daten in das Web zu exportieren, damit

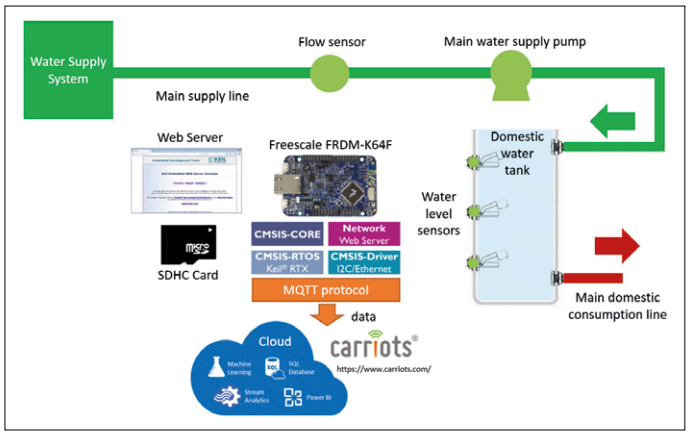


Bild 4. Das System wird via MQTT an eine Cloud angebunden. Dort stehen die Daten für verschiedene Dashboards oder IoT-Plattformen zur Verfügung.

viele potenzielle Benutzer auf diese Daten zugreifen können. Die Daten werden während der Simulation auf einer SDHC-Karte gespeichert, um das Verhalten der vielen Sensoren und Komponenten im System zu archivieren und Webseiten oder andere Systeme im Web mit Informationen zu versorgen.

In der Wolke

Über die Softwarevorlage „Data Acquisition with Web Interface“ ist es auch möglich, die Freescale-Plattform mit verschiedenen IoT-Dashboards in der Cloud zu verbinden, beispielsweise Windows Azure und Amazon Web Services, oder mit SaaS-Lösungen (software as a service), etwa der Carriots IoT-Plattform [3].
Dazu benötigt man einen Adapter oder einen Client für das MQTT-Protokoll, das das System mit verschiedenen Anbietern verbindet und Nachrichten unter einem Dach zusammenfasst. MQTT unterstützt viele mögliche Szenarien für die Nachrichtenübermittlung in IoT-Projekten. Wir versuchen zurzeit die Mosquitto-Lösung an das Freescale-Board in unserem Projekt zu koppeln (**Bild 4**). Mosquitto ist ein BSD-lizenzierter Open-Source-Broker, der die MQTT-Protokollversionen 3.1 und 3.1.1 implementiert. Weitere Informationen finden Sie unter [4].

Fazit

Unser Projekt ist in der Lage, den Wasserverbrauch eines Privathaushalts zu überwachen und anzuzeigen. Es basiert auf Wasserbehältern, Füllstandsmessern für Tanks, Durchfluss-Messgeräten und Stellgliedern für Wasserpumpen. Das Projekt verwendet eine normale Internetverbindung, um in Echtzeit Daten über die Verfügbarkeit von Wasser im Haus des Anwenders zu sammeln und die Ergebnisse für eine Webseite, für APIs, Tablets und Smartphone-Apps zur Verfügung zu stellen.
Während des Projektes konnten viele wichtige Features des Freescale FRDM-K64F genutzt werden, so dass sich das Board als sehr vielseitig für IoT-Anwendungen erwies. Es war auch eine großartige Gelegenheit, mit der Entwicklungsumgebung Keil uVision5 zu experimentieren und damit das Projekt zu konfigurieren und zu testen.
Wichtige Features, die in diesem Projekt verwendet wurden, sind:

- Log-Datei mit SDHC-Karte,
- Interrupts zum Erfassen von Daten aus dem Durchflusssensor,
- Anschlüsse von Ein- und Ausgangsventilen, LEDs und Sensoren verschiedenen Typs,
- Relais zur Steuerung externer 230-V-Verbraucher wie Wasserpumpen und
- Drucktasten, die dem Benutzer eine manuelle Eingabe ermöglichen. ⏮

(150502)

Weblinks

[1] www.nxp.com/products/software-and-tools/hardware-development-tools/freedom-development-boards/freedom-development-platform-for-kinetis-k64-k63-and-k24-mcus:FRDM-K64F

[2] www.icos.net

[3] www.carriots.com

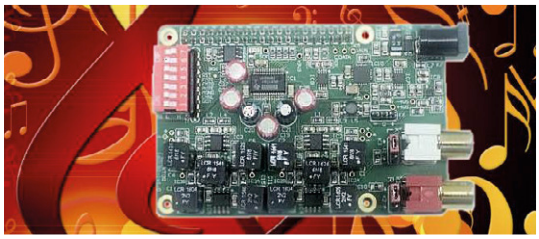
[4] http://mosquitto.org

Elektor Labs Pipeline

Viele coole Projekte werden auf der Elektormagazine.de/Labs-Website veröffentlicht. Einige sind aufwendiger als andere, doch alle sind interessant und aus dem kostbarsten Material der Welt: ~~Liebe~~ Silizium.

High-End-Audio mit Raspberry Pi

Wenn Sie nach einem kleinen, vernetzten Stand-alone-Audio-Player mit Touchscreen suchen, finden sich Geräte mit Netzwerkanschluss und ohne Touchscreen (oder ohne Netzwerk und mit Touchscreen). Oder sie haben beides, sind aber exorbitant teuer. Also, warum nicht selbst bauen? Mit einem preiswerten Raspberry Pi und einem High-End-Digital-Analog-Wandler, der 24-bit-Daten mit Abtastraten bis zu 200 kHz verarbeitet und ein integriertes 8x-Oversampling-Filter besitzt!



https://goo.gl/MdGvN9

Die Erinnerungsuhr:
Niemals wieder einen Geburtstag vergessen!

Ist es nicht peinlich, wenn Sie den Geburtstag eines geliebten Menschen verschwitzen? Gehören Sie zu den Leuten, die sich an Geburtstage und Jubiläen nur erinnern, wenn sie die Tage auf Post-its kritzeln oder jedes Jahr neu in den Kalender eintragen? Gibt es denn keinen besseren Weg? Es gibt einen! Die Erinnerungsuhr entlastet Ihr Gedächtnis und lässt Ihnen die Muße, Geburtstagsgeschenke einzukaufen oder zumindest die Jahrestagswünsche „fristgerecht“ zu übermitteln.

https://goo.gl/CRhzYq

Stromverbrauch immer im Blick

Obwohl Elektor-Projekte ohne Modifikationen gebaut und genutzt werden können, entpuppen sie sich oft auch als wahre Quellen der Inspiration. Der Power-Analyzer, den wir hier vorstellen, ist ein solches Projekt. So entwickelte sich der AC/DC-Leistungsmesser aus Elektor September 2015 zu einem hochpräzisen Instrument mit drei Kanälen und grafischer Benutzeroberfläche. Die Software wurde in MikroBasic geschrieben.

https://goo.gl/CL3pHp

Bauen Sie ein Leitfähigkeits-Messgerät!

Die Messung der Leitfähigkeit einer Chemikalie ist im Labor und im Feldversuch gang und gäbe. Ein Leitfähigkeits-Messgerät kann verwendet werden, um die Trinkwasserqualität zu überwachen, dient der Endpunkterkennung bei der Titration und ist hilfreich, wenn ein Reaktionsfortschritt überprüft werden soll. Das hier beschriebene Arduino-Shield ist in der Lage, die Konzentration von Kochsalz in Wasser von 1,5 mg/l bis etwa 25 g/l - dies entspricht dem Salzgehalt von fast destilliertem Wasser bis zu Meerwasser - mit einem typischen Linearitätsfehler von 15 % zu verfolgen..

(160346)

https://goo.gl/o0AipW

Experimentieren mit Tesla-Trafos

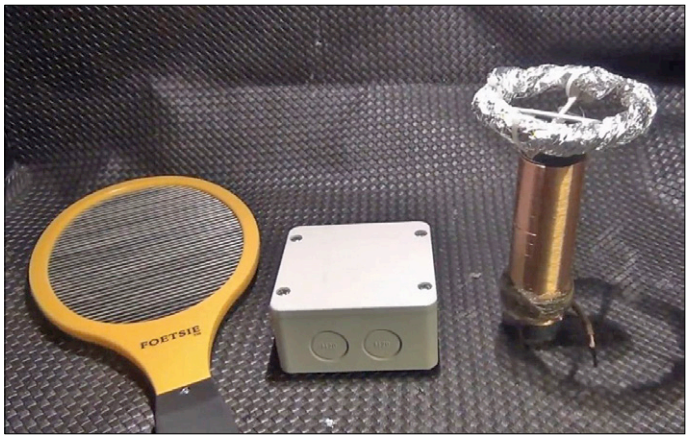
Wenig Aufwand, eindrucksvoller Effekt

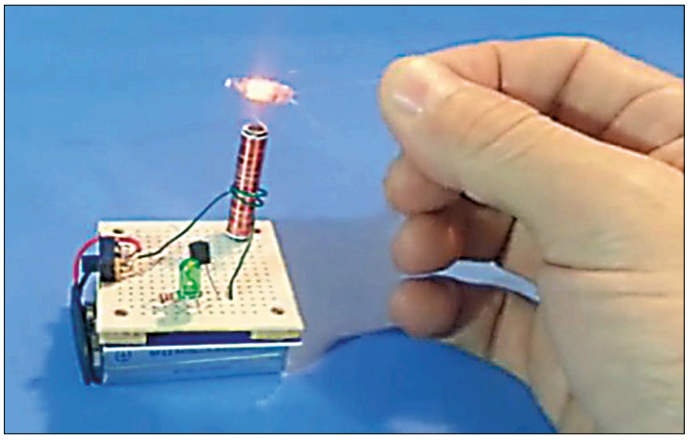
Das Phänomen der funkensprühenden Tesla-Transformatoren beeindruckt die meisten Zeitgenossen, nicht nur die Elektroniker. Die Trafo-Konstruktionen reichen von Mini bis Maxi, darunter sind imposante Aufbauten, die Funkenlängen bis zu mehreren Metern produzieren. Hier beschränken wir uns auf Versionen, die weniger lange Funken erzeugen oder lediglich Lampen drahtlos zum Leuchten bringen. Dafür lassen sie sich unkompliziert realisieren.

Von **Harry Baggen** (Elektor-Labor)

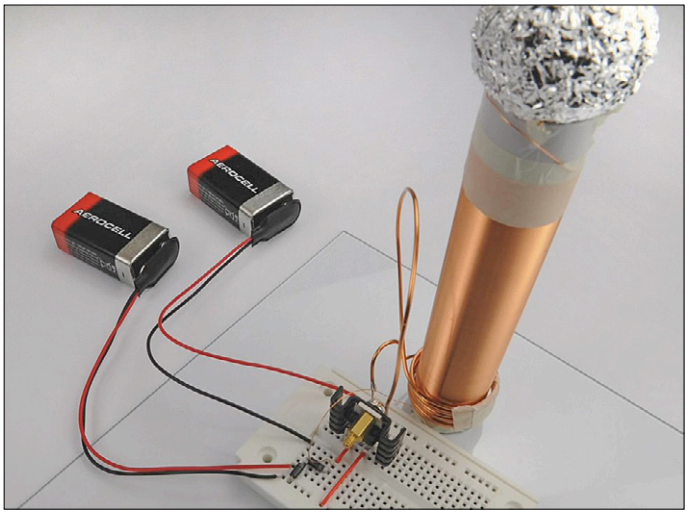
Wahrscheinlich sind Sie schon einmal einer Apparatur begegnet, die langgezogene Funken in die nähere Umgebung ausstrahlt. Ein solcher Tesla-Transformator besteht im Wesentlichen aus zwei luftgekoppelten Induktivitäten, er liefert auf der Sekundärseite eine sehr hohe Spannung. Die Spannung der Primärseite stößt den sekundären LC-Kreis an, so dass er in Resonanz gerät. Das Anstoßen wurde früher häufig durch eine Funkenbrücke auf der Primärseite bewerkstelligt. Doch weil die Funkenbrücke nicht nur viel Lärm, sondern auch elektromagnetische Störungen in der Umgebung verursacht, übernehmen heute Halbleiter- oder Röhrenschaltungen diese Funktion. Die Frequenz der primärseitigen Wechselspannung liegt typischerweise zwischen 50 kHz und 1 MHz.

Normalerweise folgt der Aufbau eines Tesla-Transformators einem bestimmten Muster. Die Sekundärseite wird durch eine langgezogene vertikale Drahtwicklung gebildet. Auf der oberen Seite befindet sich eine kapazitive Elektrode, meistens in Form eines Torus. Die Durchschlagspannung der Luft bewegt sich zwischen 1 kV und 3 kV pro Millimeter, sie hängt unter anderem von der Form der Elektroden ab. Spitze Elektroden fördern den Funkenüberschlag, während Elektroden in Kugel-





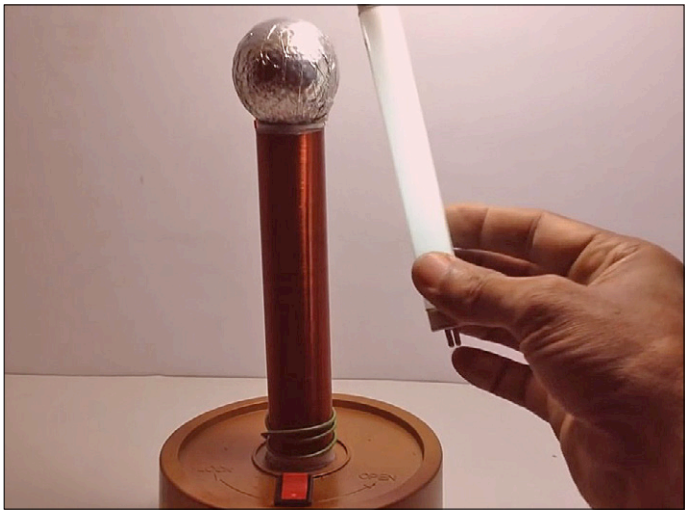
form ihn eher bremsen. Wenn ein Millimeter Luftstrecke mit 3 kV durchschlagen wird, muss die Spannung 300 kV betragen, um 10 cm lange Funken zu erzeugen!
Über die elektrotechnischen Hintergründe von Tesla-Transformatoren gibt am ausführlichsten die englischsprachige Wikipedia Auskunft [1]. Eine recht praxisnahe Beschreibung ist auf der Website von HVtesla [2] zu finden, illustriert mit eindrucksvollen Abbildungen diverser existierender Tesla-Transformatoren. Der *Tesla Coil Design, Construction and Operation Guide* von Kevin Wilson [3] ist ebenfalls ein Tipp für Leser, die etwas mehr erfahren möchten.

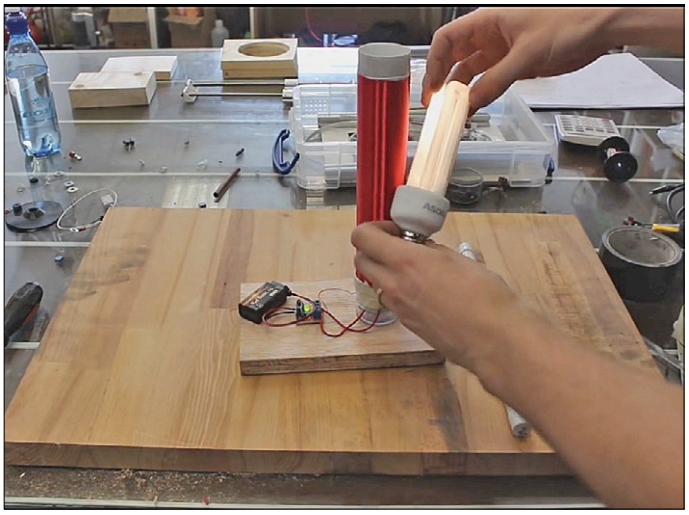


Nicht so kompliziert

Der Bau eines ausgewachsenen Tesla-Transformators, der meterlange Funken sprüht, ist nicht ganz einfach. Zur mechanischen Konstruktion kommt die Elektrik oder Elektronik hinzu, sie muss für Hochspannung bei einiger Leistung ausgelegt sein. Doch wenn auf dem Aktionsplan lediglich einführende Experimente stehen, genügt eine weniger aufwändige Herangehensweise.
Um eine Hochspannung zu generieren, ist das Zweckentfremden eines preiswerten Geräts, das mit Hochspannung arbeitet, eine bequeme Methode. Dazu zählen zum Beispiel das Blitzgerät einer Einwegkamera oder eine elektrische Insektenklatsche. In dem zweiteiligen Video, das unter dem Titel *How to make a small Tesla Coil* bei YouTube steht [4], wird das Umfunktionieren einer Insektenklatschen-Schaltung zu einem Mini-Tesla-Transformator demonstriert. Trotz der simplen Konstruktion sind erstaunliche Funkenlängen produzierbar.
Das gleiche Ziel lässt sich natürlich auch mit einzelnen Bauteilen erreichen. Die meisten Schaltungen kommen mit nur einem Transistor aus, ein Beispiel mit bebilderter Bauanleitung steht auf der Website von *Learn to discover* [5]. Die erreichbare Funkenlänge ist zwar nicht überwältigend, doch was hier deutlich wird, ist das Prinzip.
Solche wirklich simplen Konstruktionen werden häufig auch *Slayer exciter* genannt. Damit sind Schaltungen gemeint, die so starke elektromagnetische Felder erzeugen, dass in der Nähe befindliche Fluoreszenz- oder Glimmlampen aufleuchten. Das Ziel ist hier nicht unbedingt das Produzieren elektrischer Funken. Einen Besuch ist auch die Seite *Miniature Tesla Coil* auf dem Portal *Hackaday* wert [6]. Das aktive Bauelement des Spannungserzeugers ist ein simpler Kleinsignaltransistor 2N2222, und die bleistiftdünne Sekundärwicklung ist eine recht unscheinbare Konstruktion. Trotzdem werden Fluoreszenz- und Leuchtstofflampen im Nahfeld drahtlos zum Leuchten angeregt.
Auf YouTube sind überraschend viele Videos versammelt, in denen es um Experimente der Kategorie *Slayer exciter* geht. Die vier Beispiele [7] haben nicht nur Schauwert, sie sind auch lehrreich. ⏮

(160352)gd





Weblinks

[1] https://en.wikipedia.org/wiki/Tesla_coil

[2] www.hvtesla.com/

[3] www.teslacoildesign.com/

[4] www.youtube.com/watch?v=QDZnCOLZ394
www.youtube.com/watch?v=X2PrPHgOy04

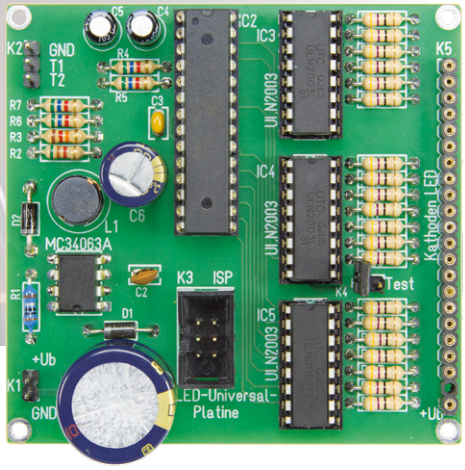
[5] https://learntodiscover-science.blogspot.nl/2016/10/simple-tesla-coil.html

[6] http://hackaday.com/2014/12/22/micro-tesla-coil-makes-a-perfect-stocking-stuffer/

[7] www.youtube.com /watch?v=MwG-EzVEE1I
www.youtube.com/watch?v=3zjnX41K1pE
www.youtube.com/watch?v=LHCXqhhxGqA
www.youtube.com/watch?v=gSaGzhdI-QM

Treppen-beleuchtung mit LEDs

Ein Controller für viele Leuchtdioden



Von **Peter Grundmann**

Haustechnik der besonderen Art: Die kleine Controllerschaltung steuert LED-Ketten, die das Treppenhaus in einer eleganten und faszinierenden Weise illuminieren. Da wird der Aufstieg zum Erlebnis!

Ein Bekannter berichtete mir von einer effektvollen Treppenbeleuchtung: Mit einem Taster am unteren Treppenende wurden die Stufen langsam und nach oben heller werdend beleuchtet. Am oberen Ende konnte man die Beleuchtung wieder ausschalten, unten beginnend und genauso langsam. Natürlich funktionierte das Ganze auch umgekehrt. Sehr schön, aber das Steuergerät kostete mehrere hundert Euro, das war jenseits von Gut und Böse. Da ich mich aber seit geraumer Zeit mit Atmel-Controllern beschäftigte, fragte ich mich, ob es nicht auch preiswerter gehen könnte.

Gedanken vorab

Da eine Treppe in einem Wohnhaus selten mehr als 20 Stufen hat und zum Ein- und Ausschalten auch nicht mehr als zwei Taster nötig sind, habe ich als Controller einen ATmega8 ausgewählt. Da keine serielle Schnittstelle und kein Quarz nötig sind, reichen die Ports locker aus.
Jede Stufe soll von einer Kette in Reihe geschalteter LEDs beleuchtet werden, im Bedarfsfall auch von mehreren solcher Ketten parallel. Klar, dass weder Spannung noch Strom eines Mikrocontrollerausgangs dafür ausreichen. Es braucht leistungsfähige Treiber, die in der Lage sind, die entsprechende Spannung zu schalten.

Diese Aufgabe können die sieben in einem DIP-Gehäuse zusammengefassten Treiber der Serie ULN2003 leicht übernehmen. Sie enthalten bereits einen Widerstand im Eingang, können also direkt mit dem Ausgang des Controllers verbunden werden. Sie sind mit Darlingtontransistoren aufgebaut, die bis zu 500 mA und 50 V verkraften, viel mehr also, als hier nötig wäre. In der Schaltung in **Bild 1** sind außer dem Controller und den drei Leistungstreibern noch der Anschluss K2 für die beiden Taster und die Stromversorgung zu sehen.
Damit die Schaltung sicherheitstechnisch völlig unbedenklich gestaltet werden kann, soll die Stromversorgung aus einem Steckernetzteil erfolgen. Solche Netzteile, die beispielsweise von einem ausrangierten Router oder ähnlichen Geräten aufbewahrt werden, liefern meist Gleichspannungen von 12...20 V. Für den Prozessor sind nur etwa 4 V erforderlich, so dass ein Längsregler eine durchaus nennenswerte Leistung aufnehmen müsste. Zwar ist der Eigenstrombedarf des Controllers gering, rechnet man jedoch pro eingeschaltetem Ausgang ein knappes Milliampere hinzu, so wird klar, dass die kleinen L-Typen der dreibeinigen ICs schnell an ihre Grenze stoßen würden. Entweder müsste man einen großformatigen TO220-Regler einsetzen oder einen Schaltregler wie den MC34063A, der mit wenigen externen Bauteilen die Aufgabe effizienter lösen kann.

Aufbau der Schaltung

Ein erster Aufbau erfolgte natürlich auf einer Lochrasterplatine. Die Treiber im DIP-Gehäuse waren recht schnell und einfach zu verdrahten und auch die dem

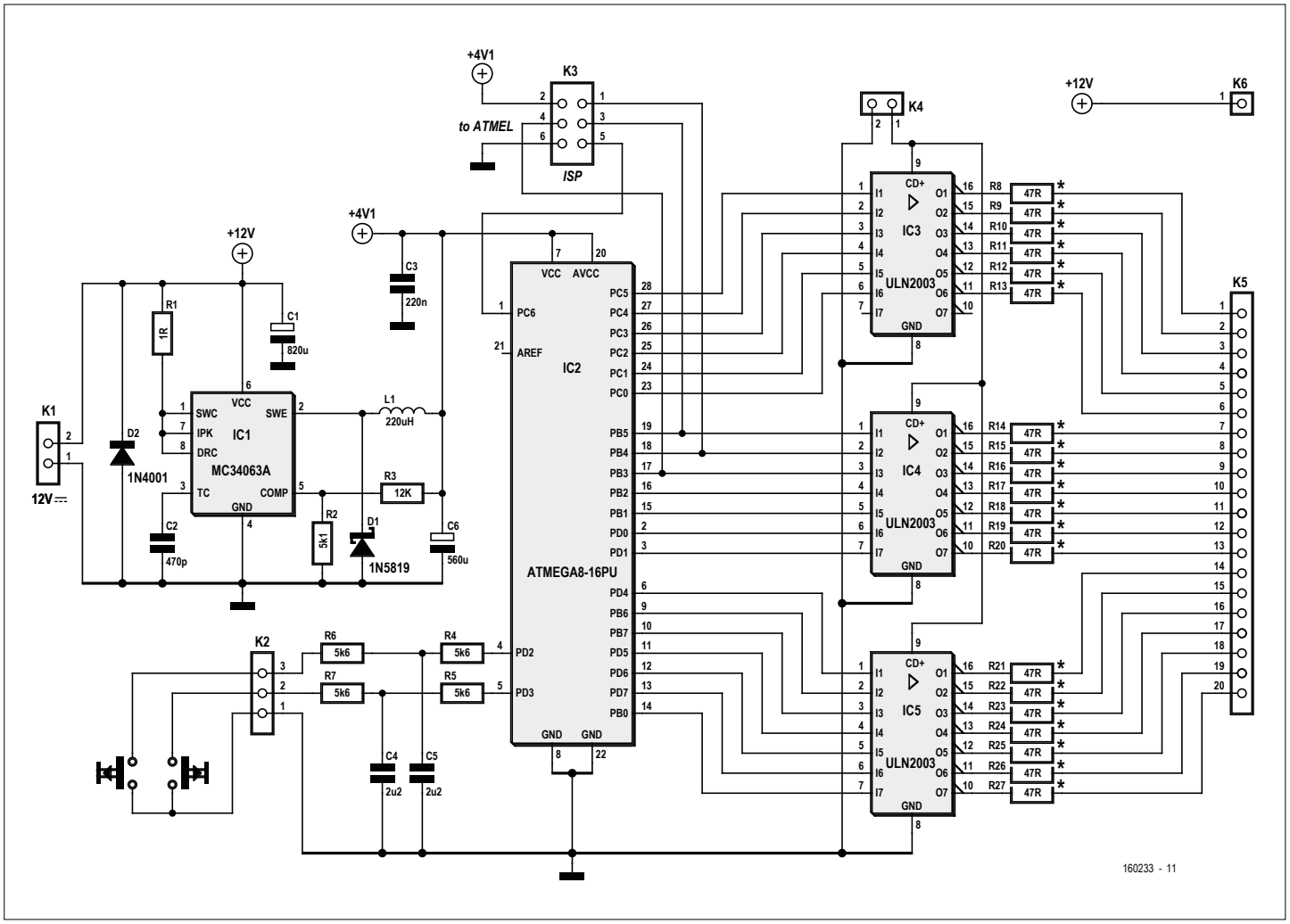


Bild 1. Ein Atmel-Controller mit Muskeln.

Datenblatt entnommene Schaltung des Step-down-Konverters war schnell und kompakt verwirklicht. Die Anschlüsse zum Programmieren des Controllers liegen jedoch so ungünstig, dass eine kleine doppelseitige Platine erhebliche Vorteile bietet. Diese Platine enthält noch ein paar Bauteile mehr als für den Betrieb eigentlich erforderlich wären. So soll die Diode D2 einfach nur eine „Lebensversicherung" für den Fall darstellen, dass man die Betriebsspannung falsch anschließt. An den Tastereingängen sind RC-Glieder geschaltet, die bei mehreren Metern Leitungslänge eventuell eingestreute Störsignale unterdrücken. Und an den Ausgängen wurde zusätzlich eine Horde von Strombegrenzungswiderständen platziert.

Die ULN2003 sind für das Schalten induktiver Lasten mit Schutzdioden ausgestattet. Deren gemeinsam herausgeführte Kathode kann über einen Jumper mit GND verbunden werden. Damit lassen sich, unabhängig vom Controllerprogramm, alle LEDs zu Testzwecken einschalten, um zum Beispiel die Verdrahtung zu testen.

Das Programm in Bascom

Das Programm wurde mit Bascom realisiert (Download unter [1]). Da ich über die kostenpflichtige Version ohne Code-Begrenzung verfüge und Flash-Speicher reichlich vorhanden ist, habe ich die Programmgröße nicht optimiert. Das Programm müsste aber auch mit der Demo-Version des Compilers zu bearbeiten sein.

Der einfachen Nachvollziehbarkeit halber sind die Portpins einzeln deklariert. Die einzelnen Ein- und Ausgänge werden über Alias-Namen angesprochen, was die Portierung auf andere Controllertypen oder ein geändertes Routing erleichtert. Eine kleine Herausforderung war das pulsbreiten-modulierte Ein- und Ausschalten der LEDs. Die eingebauten PWM-Funktionen sind ja nur an wenigen Portpins nutzbar, konnten also kaum helfen. Aber eigentlich geht das auch „mit der Hand": Eine Variable `Pwm_hilf` zählt ständig von 1 bis 254 und beginnt dann wieder bei 1. Eine Byte-Variable pro LED beziehungsweise Treppenstufe stellt den Helligkeitswert dar. Ist sie kleiner als `Pwm_hilf`, bleibt die LED ausgeschaltet, ist sie größer, so ist die LED eingeschaltet. Bei einem Helligkeitswert 0 ist die LED also ständig dunkel, bei Werten von 1 bis 254 nimmt das Einschaltverhältnis zu, bis beim Wert 255 ein Tastverhältnis von 100 % erreicht wird. Es bleibt dann nur noch, die Vergleichsroutine in einen Interrupt zu packen, der in angemessenen Zeitabständen aufgerufen wird. Angemessen heißt, dass alles in einer für das menschliche Auge unsichtbaren schnellen Folge geschehen soll.

Beim Betätigen der Taster müssen die Helligkeitswerte der einzelnen Stufen mit einer Verzögerung hoch- beziehungsweise heruntergezählt werden. Dies wird

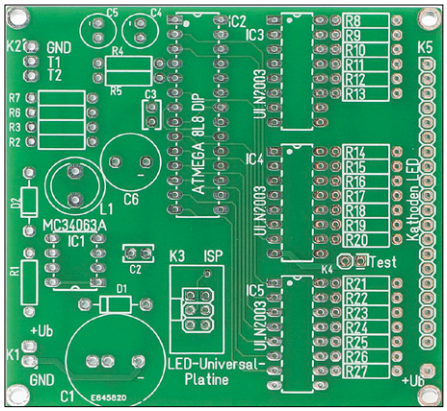


Bild 2. Die Platine ist übersichtlich und leicht zu bestücken.

einfach über ein paar Bits gesteuert und in vier Schritten realisiert. Wenn Sie die Änderungsgeschwindigkeit eigenen Vorstellungen anpassen möchten, können Sie dies leicht durch Variation der Werte der Variablen `Verzoegerung` beziehungsweise `Verzoegerung2` verwirklichen.

Energieeffizienz

Eine solche Beleuchtungseinrichtung sollte natürlich sparsam arbeiten. Bei etwa 12 V Eingangsspannung verbraucht der Schaltregler circa 2,5 mA, ist der Controller in den Sockel eingesteckt, kommen nochmals etwa 3 mA hinzu. Versuche mit unterschiedlichen Ausführungen des ATmega8 ergaben keine nennenswerten Unterschiede. Bei 36 mW Leistungsaufnahme schienen aber auch weitere Versuche mit den unterschiedlichen Sleep-Modi nicht besonders sinnvoll. Im eingeschalteten Zustand erhöht sich der Eigenstromverbrauch der Schaltung auf etwas mehr als 14 mA, dazu muss man natürlich den Betriebsstrom der LEDs addieren. Die „energetische Katastrophe“ stellen dann natürlich die leider unverzichtbaren Vorwiderstände der LED dar. Mit ihren rechnerischen fast 2,5 W Abwärme verursachen sie im Betrieb die größten Verluste. Da sich diese Verluste aber nur beim doch vergleichsweise seltenen Betrieb ergeben, dürfte der Energiehunger dieser Schaltung zu tolerieren sein.

Aufbau und Bestückung

Bild 2 zeigt mein endgültiges Platinenlayout, das als Target 3001!-Dateien von der Elektor-Projektseite [1] heruntergeladen werden kann. Der Aufbau der Schaltung auf der übersichtlich gestalteten Platine sollte keine Schwierigkeiten bereiten. Alle Teile sind handelsüblich, nur sollte man vor dem Bestellen der Vorwiderstände über deren Wert und die Belastbarkeit nachdenken. Die angegebenen 47 Ω und ¼ W Belastbarkeit sind für 12 V Betriebsspannung und insgesamt neun weiße LED je Stufe ausgelegt. Dabei werden je drei LEDs hintereinandergeschaltet und drei solcher Ketten parallel an einen Ausgang angeschlossen. Die Leuchtdioden sollten alle aus einer Charge stammen, damit keine größeren Unterschiede in den Flussspannungen zu unterschiedlichen Helligkeiten führen.
Bei der Dimensionierung der Vorwiderstände ging ich zunächst von den Werten im Datenblatt der LED aus. Experimente mit den gelieferten Leuchtdioden zeigten aber eine deutlich geringere Flussspannung als die Werte, die dem entsprechenden Diagramm im Datenblatt zu entnehmen waren. Es empfiehlt sich also, die Widerstandswerte experimentell zu ermitteln.
Es versteht sich fast von selbst, dass beim Einsatz fertiger LED-Streifen mit eingebauten Vorwiderständen für 12 V Betriebsspannung die Vorwiderstände in der Schaltung einen Wert von 0 Ω haben sollten, also eine Drahtbrücke einzulöten ist. An den Darlingtons in den ULN-Treibern fällt im eingeschalteten Zustand eine Sättigungsspannung von ungefähr 0,8...1 V ab. Dies muss man berücksichtigen und die Betriebsspannung unter Umständen anpassen, um die gewünschte Helligkeit zu erreichen.

Wie man die zahlreichen Anschlüsse herstellt, ist Geschmackssache. Für Experimente haben sich Steckkontakte bewährt, bei fester Installation darf man die Kabel auch einfach in die Löcher löten.

Weitere Ideen

Die Platine kann natürlich auch in anderen Projekten verwendet werden, schließlich ist alles vorhanden, was man zur Ansteuerung von LEDs braucht. Laut Datenblatt kann der Schaltregler mit Eingangsspannungen bis zu 40 V fertig werden. Eine solche Anwendung würde natürlich auch einen entsprechend spannungsfesten Ladeelko C1 verlangen. Der Kreativität sind allenfalls Grenzen durch die 8 K Flashspeicher im ATmega8 gesetzt. Wem das nicht genug ist, der findet im ATmega328 einen pinkompatiblen, aber leistungsfähigeren Ersatz.

(160233)

Weblink

[1] www.elektormagazine.de/160233 (Bascom-Quellcode und Target-Platinenlayout-Dateien)

Über den Autor

Peter Grundmann (Jahrgang 1956) hat sich schon in früher Jugend mit der Elektrotechnik beschäftigt. Das Betätigungsfeld war meist die Modelleisenbahn. Später absolvierte er eine kaufmännische Ausbildung, inzwischen arbeitet er in der Software-Entwicklung.
Der Hobby-Elektronik ist er jedoch treu geblieben. Nach analogen und digitalen Schaltungen bildet inzwischen die Mikrocontroller-Programmierung die Hauptbeschäftigung. Inzwischen kann der Autor auf einen gewissen Erfahrungsschatz zum Beispiel bei der Realisierung von Beleuchtungseffekten und der Anwendung von Servos zur Ausführung von Bewegungseffekten im Modellbahnbereich zurückgreifen. Diese sind inzwischen auch in kommerzielle Produkte eingeflossen.

Gerne tauscht Peter Grundmann Erfahrungen aus, die Kontaktaufnahme per E-Mail ist über peter.grundmann@meier-modellbau.de leicht möglich.

Niedrige Geschwindigkeit, hohe Reichweite

Von **Robert Lacoste** (F)

In meinem letzten Beitrag habe ich gezeigt, dass die Empfindlichkeit eines Funkempfängers von drei Parametern abhängt: Der Qualität der elektronischen Komponenten, der Übertragungsgeschwindigkeit und den Eigenschaften des verwendeten Demodulators. Diesmal gehen wir den Schritt von der Theorie zur Praxis, indem wir uns LoRa anschauen, eine Übertragungstechnik, die von Semtech Semiconductors entwickelt wurde. Ein späterer Beitrag wird sich mit LoRaWAN beschäftigen, ein Protokoll, das auf LoRa aufsetzt.

LPWAN?

Lassen Sie uns mit einer allgemeinen Überlegung beginnen. Bekanntlich geht die Fachwelt davon aus, dass in den kommenden Jahren und Jahrzehnten Milliarden IoT-Geräte weltweit miteinander in Verbindung treten. Wenn Sie ein solches IoT-fähiges, selbstverständlich drahtloses Gerät entwickeln müssten, wie würden Sie dieses Gerät mit dem Internet verbinden? Kurz zusammengefasst gibt es dafür drei Lösungen (**Bild 1**).

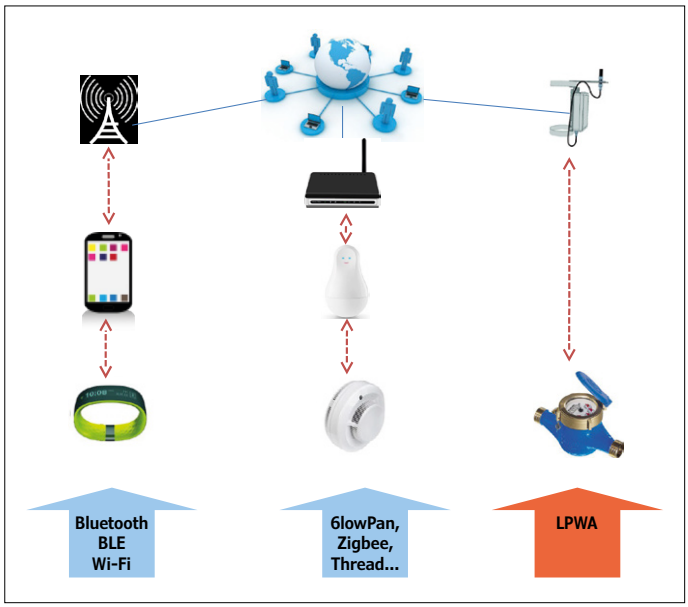


Bild 1. Die drei relevanten Netzwerk-Familien, die IoT-Geräte an das Internet koppeln können.

Weg 1 ist naheliegend: Man nutzt ein kommunikatives Gateway, das in entwickelten Ländern fast jeder Mensch in der Tasche trägt: Das Smartphone. Ihr neu entwickeltes IoT-Gerät kann drahtlos über Bluetooth, Bluetooth Low Energy oder WLAN an das Smartphone angebunden werden, und das Smartphone stellt die Verbindung mit dem Internet her. Die Pluspunkte dieser Lösung liegen auf der Hand: Übersichtliche Struktur, niedriger Energiebedarf und erfreulich niedrige Kosten. Nur das Smartphone ist unverzichtbar.

Die zweite Lösung ersetzt das Smartphone durch ein stationär installiertes Gateway. Wenn Ihr IoT-Gerät mit einer WLAN-Schnittstelle ausgestattet ist, kann Ihr Internet-Router die Funktion des Gateways übernehmen. Doch zur Verbindung mit dem Gateway sind auch andere Protokolle möglich, zum Beispiel Zigbee, Zwave, 6lowPan oder Thread. Die Kosten bewegen sich auf niedrigem Niveau, ein Smartphone ist entbehrlich, allerdings ist das Installieren eines Gateways nur selten ganz einfach. Hinzu kommt, dass die Anzahl der benötigten Gateways bei räumlich umfangreichen Projekten schnell steigt, denn die Übertragungsstrecken haben nur kurze Reichweiten.

Die dritte Lösung besteht darin, lokale Gateways vollständig auszuklammern. An ihre Stelle tritt eine Technologie, die eine große räumliche Reichweite hat. Wenn weite Distanzen überbrückbar sind, ist die Kommunikation mit einem fernen Gateway möglich. Ein solches abgesetztes Gateway kann eine Straße, einen Häuserblock oder eine ganze Ortschaft erfassen. Für diese Lösung bietet sich wegen des flächendeckenden Netzausbaus die Nutzung eines bestehenden Mobilfunknetzes an (beispielsweise 2G oder 3G), der Ressourcenbedarf und die Kosten sind jedoch beträchtlich. In den letzten Jahren hat sich diese Situation dank innovativer Entwicklungen deutlich gewandelt. So

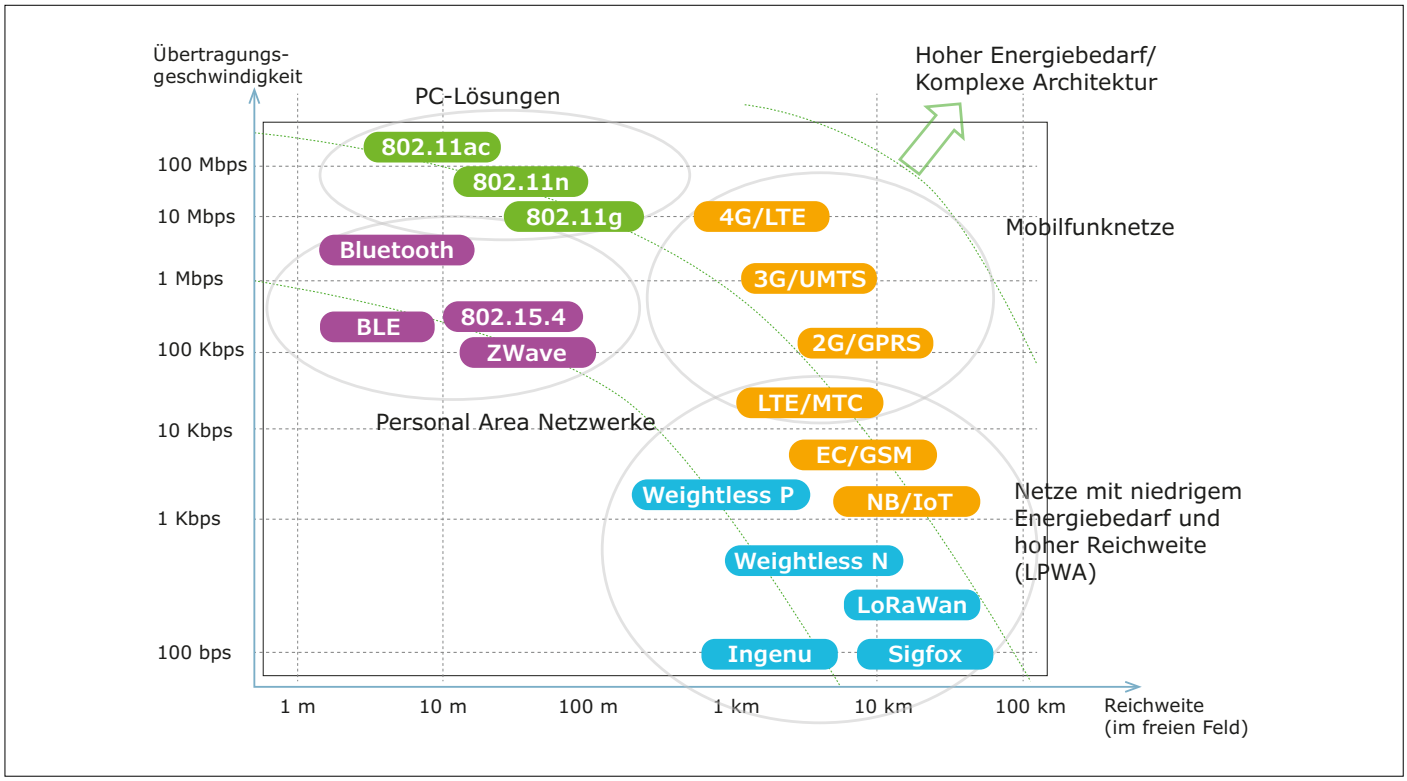


Bild 2. Übersicht über Verfahren der drahtlosen Datenübertragung. LPWA-Funknetze erkaufen Reichweite mit niedriger Geschwindigkeit.

entstanden unter der Sammelbezeichnung LPWAN (*Low Power Wide Area Network*) neue, zukunftweisende Netzwerke. Die zugehörigen Übertragungsstrecken sind in der Lage, Dutzende Kilometer zu überbrücken. Trotzdem hält sich der Energiebedarf der Infrastruktur in engen Grenzen.

Bitgeschwindigkeit!

In **Bild 2** ist dargestellt, dass die etablierten drahtlosen Übertragungstechniken, beispielsweise WLAN oder Bluetooth, mit hohen Bitgeschwindigkeiten arbeiten, so dass die Reichweiten zwangsläufig niedrig sind. LPWA-Netzwerke müssen andere Kriterien erfüllen: Sie sollen weite Distanzen überbrücken, aber trotzdem mit wenig Energie auskommen. Diese Forderungen sind nur erfüllbar, wenn die Übertragungsgeschwindigkeit drakonisch reduziert wird. Zu den neuen Entwicklungen auf diesem Gebiet gehört neben Sigfox, Ingenu oder Weightless auch LoRa. Auffallend ist bei dem Trend, dass auch die Mobilfunk-Dienstleister zu den Mitstreitern gehören. Parallel zu 4G, dessen Geschwindigkeit so hochgezüchtet wurde, dass HD-Videos auf Smartphones betrachtet werden können, entsteht eine eigene IoT-Netzwelt. Ein Beispiel ist die Gruppe der NB-IoT-Netzwerke (*Narrow Band Internet of Things*), sie arbeiten mit drastisch reduzierten Geschwindigkeiten.

Das Herabsetzen der Übertragungsgeschwindigkeit einer digitalen Funkverbindung ist nicht trivial. Angenommen die Geschwindigkeit beträgt 10 kbps, und zum Einsatz kommt die Modulationsart FSK (*Frequency Shift Keying*). Bei dieser Modulationsart springt die Trägerfrequenz abhängig vom Modulationsinhalt (logisch 0 oder 1) um einen Mittelwert, im angenommenen Fall im Bereich ±10 kHz. Wenn die Geschwindigkeit beispielsweise auf 10 bps bei unveränderten Modulationseigenschaften herabgesetzt werden soll, muss die Frequenzverschiebung auf ±100 Hz reduziert werden. Dies ist eine Klippe, die nicht leicht umschiffbar ist: Der Empfänger muss extrem frequenzstabil arbeiten! Bei der Sendefrequenz 868 MHz ist der Frequenzhub ±100 Hz gleichbedeutend mit der Abweichung ±0,1 ppm (*parts per million*), sie liegt um Größenordnungen niedriger als die Toleranz eines durchschnittlichen Quarzes. Diese Ultra-Breitband-Technik wurde tatsächlich realisiert, sie ist bei Systemen von Sigfox im Einsatz. Die Praxistauglichkeit wird dort durch hochkomplexe technische Kunstgriffe auf der Empfängerseite erkauft.

Das LoRa-Konzept

LoRa ist ein alternativer Weg zu dem Ziel, die Übertragungsgeschwindigkeit stark herabzusetzen, ohne mit hoch frequenzkonstanten Signalen arbeiten zu müssen. Kurz nebenbei bemerkt: LoRa wurde 2009 vom französischen Start-Up-Unternehmen Cycleo konzipiert und entwickelt. Im Jahr 2013 übernahm der Halbleiterhersteller Semtech das noch junge Unternehmen.

Bei LoRa ist die Modulationsbandbreite immer hoch, sie liegt typisch zwischen 125 kHz und 500 kHz. Die zu übertragenden Bits werden jedoch in einem Verfahren codiert, das die Geschwindigkeit reduziert, ohne die Bandbreite ändern zu müssen. Außerdem werden komplexe Methoden angewandt, unter anderem für die Fehlerkorrektur, die sich auf die Empfindlichkeit des Empfängers steigernd auswirken. Bereits im vorangegangenen Beitrag [1] haben wir festgestellt, dass die Empfindlichkeit eines Empfängers bei gleicher Bitgeschwindigkeit und Empfänger-Architektur unabhängig von der Modulationsbandbreite ist. Deshalb können Breitbandsysteme wie LoRa einige starke Pluspunkte für sich verbuchen: Signale lassen sich mit herkömmlichen Mitteln frequenzkonstant halten, die Störempfindlichkeit liegt niedriger als bei Schmalbandsyste-

men, die Systeme sind anpassbar und flexibel. Dem stehen leider auch einige Schwachpunkte gegenüber: Im Empfänger müssen komplexe digitale Operationen mit spezialisierten Chips vorgenommen werden, und bei den nutzbaren Funkfrequenzen drohen mittelfristig deutliche Engpässe. Das zuletzt genannte Problem ist jedoch gegenwärtig noch nicht aktuell.

CSS, was ist CSS?

Im so genannten Schichtenmodell ist LoRa das, was dort *Physical Layer* genannt wird. Das bedeutet, dass LoRa die Bitströme digitaler Nachrichten so verpackt, dass sie möglichst effizient drahtlos übertragen werden können. Auf der Empfängerseite führt LoRa den inversen Prozess durch. Die an Anwendungen gebundenen Funktionen wie das Einfügen von Adressen oder das Auswerten von Empfangsquittungen gehören nicht dazu. Das sind Aufgaben von Protokollen wie beispielsweise LoRaWAN, die in höheren Schichten angesiedelt sind.

Wie schon erwähnt, gehört zu den Stärken von LoRa seine hohe Flexibilität. Die Trägerfrequenz ist zwischen 137 MHz und 1020 MHz wählbar, in Europa nutzen die meisten Anwendungen das 868-MHz-Band. Ferner lassen sich drei Parameter einstellen, die ebenfalls Einfluss auf die Übertragungsgeschwindigkeit haben (**Bild 3**). Zuerst ist die **Modulationsbandbreite** (*Band Width*, BW) zu nennen, sie kann zwischen 7,8 kHz und 500 kHz liegen. Höhere Bandbreiten lassen grundsätzlich höhere Geschwindigkeiten zu. Als nächstes lässt sich der so genannte **Spreizfaktor** (*Spreading Factor*, SF) im Bereich 6...12 verändern. Das Erhöhen dieses Faktors um 1 hat eine Halbierung der Geschwindigkeit zur Folge. Wählbar ist auch die Effizienz der **Fehlerkorrektur** (*Error Correction*, EC), sie ergänzt immer 4 Nutzbits um 0 bis 4 Fehlerkorrekturbits.

Das klingt recht beeindruckend, doch wie sehen die Details aus? Hier taucht das Problem auf, dass die Details der LoRa-Modulation von Semtech bisher nicht freigegeben wurden. Semtech hat lediglich dem speziellen Modulationstyp einen Namen gegeben: Er trägt die Bezeichnung *Chirp Spread Spectrum*, abgekürzt CSS [2]. Die Analyse eines LoRa-Signals mit einem herkömmlichen Spektrumanalyser ergibt keinen näheren Aufschluss zu CSS. Sichtbar ist lediglich eine Art „Hochebene" mit der Ausdehnung der Bandbreite (BW), die sich um die Trägerfrequenz legt. Das Signal besteht offensichtlich aus einem Träger, dessen Frequenz zeitveränderlich ist, ungefähr wie bei FM. Um etwas tiefer hinter die Kulissen zu schauen, ist ein Spektrumanalyser nötig, der den Frequenzverlauf über eine definierte Zeit messen und darstellen kann. Der Autor konnte auf ein Gerät mit dieser Funktion zurückgreifen, es war ein Spektrumanalyser RSA5106 von Tektronix. Das Ergebnis der Messungen ist in **Bild 4** wiedergegeben. Der Darstellung der Frequenz als Funktion der Zeit lässt sich entnehmen, dass die Frequenz fortlaufend mit konstanter Geschwindigkeit über die gesamte Bandbreite gewissermaßen „schwingt". Außerdem sind Sprünge in dieser Bewegung sichtbar, die auf das Dekodieren der übertragenen Bits hindeuten. Nun ist die Bezeichnung *Chirp Spread Spectrum Frequency Modulation* erklärlich: Wenn das „schwingende" Signal in den Audiobereich transponiert wird und einen Lautsprecher steuert, ist ein zirpendes Geräusch hörbar.

Zwar wurden die Details der LoRa-Modulation nicht offen gelegt, doch sie wurden von Experten weitgehend enträtselt. In **Bild 5** ist global eine Codierung mit folgenden Parametern dargestellt: BW = 125 kHz, SF = 12, Korrekturcode 5/4. Mit BW = 125 kHz beträgt die Frequenzschwebung ebenfalls 125 kHz, dies ist die

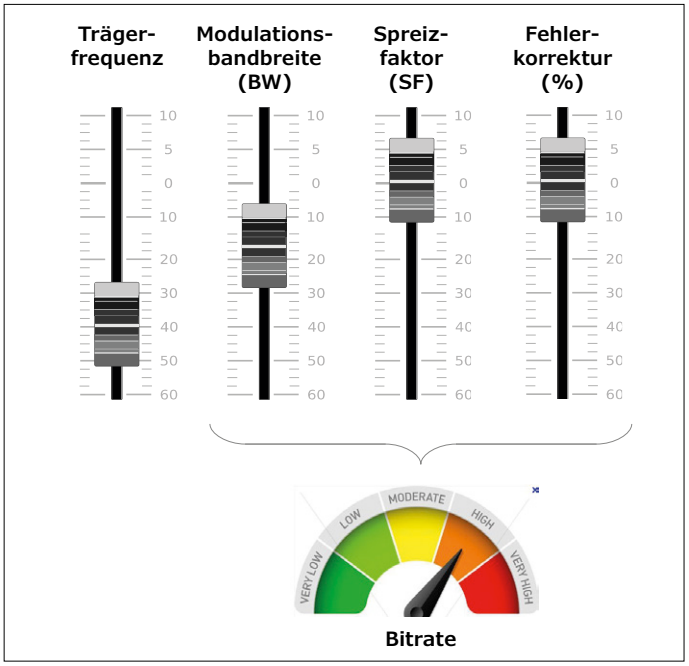


Bild 3. Bei LoRa können vier Parameter eingestellt werden, um die Reichweite und Geschwindigkeit zu beeinflussen.

Amplitude an der y-Achse des Diagramms *Frequenz = f(t)*. Daraus geht hervor, dass ein Frame mit einem Vorspann beginnt, der aus vollen Frequenzschwebungen besteht. Dann folgen einige „inverse" Schwebungen, wobei die Frequenz abnimmt statt zunimmt. Sie sind das Synchronisationssignal, das dem Empfänger den Beginn der Nutzinformation signalisiert.

Die Bits sind in Paketen gruppiert, die *Symbole* genannt werden. Wenn SF = 12 gewählt wurde, umfasst jede Gruppe genau 12 Bit. Ein Symbol ist dann eine Zahl zwischen 0 und 4095 (= $2^{12} - 1$), es wird in einem einzigen Schritt übertragen. Im Diagramm sind in Abständen von 32,8 ms Sprünge in der Reihe der Frequenzschwebungen sichtbar. Der Wert stimmt mit der Dauer eines Symbols in LoRa mit BW = 125 kHz und SF = 12 überein, er ergibt sich aus der Beziehung 2^{12}/125.000 Hz = 0,0328 s. Mit

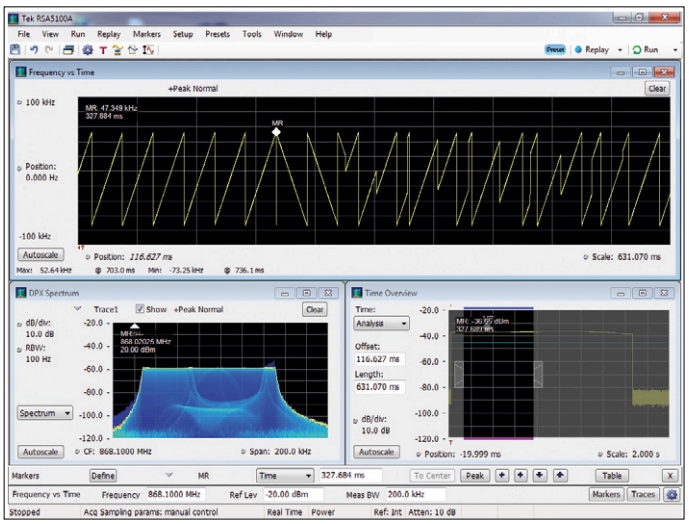


Bild 4. Ein LoRa-Frame, dargestellt von einem Real-time-Spektrumanalyser, hier ein Tektronix RSA5106.

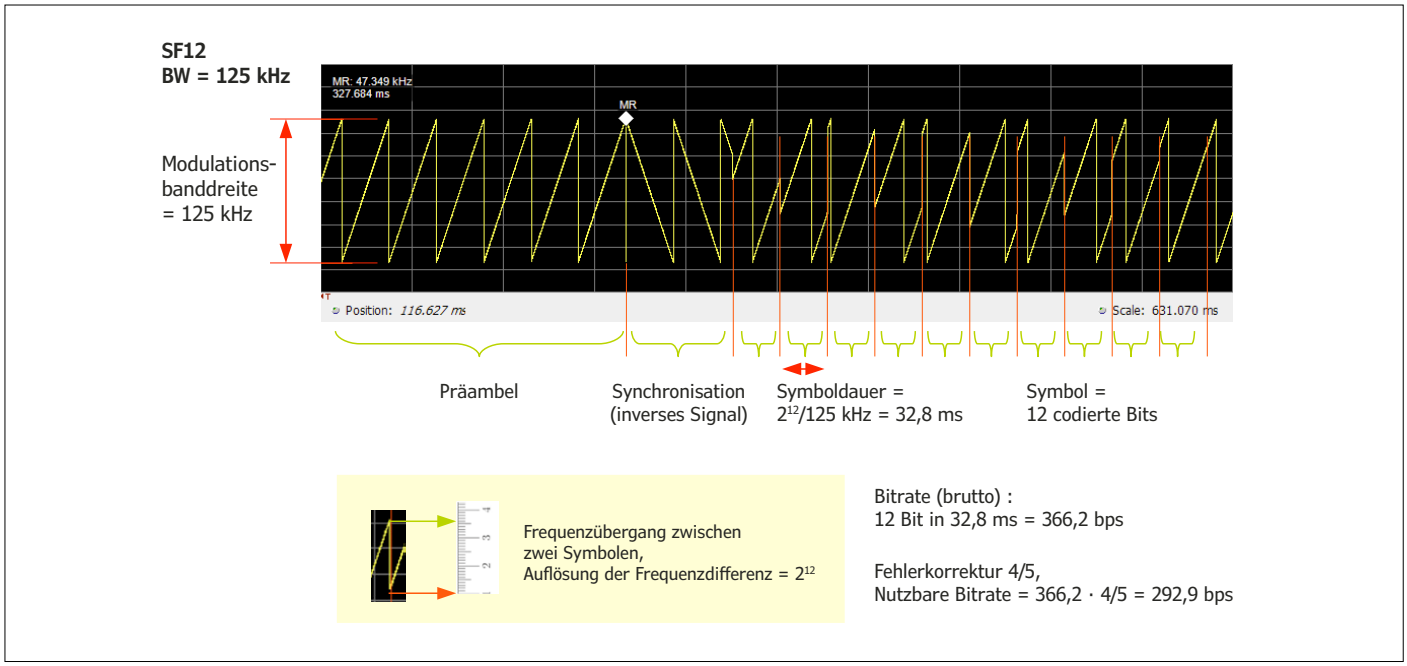


Bild 5. LoRa ist enträtselt!

diesen Einstellungen überträgt LoRa in Abständen von 32,8 ms ein aus 12 Bit bestehendes Symbol, was der Übertragungsgeschwindigkeit 366,2 bps entspricht. Wenn dieser Konstellation für jeweils 4 Bit ein Fehlerkorrekturbit hinzugefügt wird, sinkt die nutzbare Geschwindigkeit auf 292,9 bps.

Hier stellt sich die Frage, wie ein aus 12 Bit bestehendes Symbol in einem einzigen Frequenzsprung codiert werden kann. Die Antwort ist einfach, zumindest in der Theorie: Dazu wird die Frequenzschwebung um 125 kHz in 2^{12} = 4096 Abschnitte unterteilt, wobei jeder Abschnitt gleich 125.000 / 4096 = 30,52 Hz ist. Wenn das Symbol aus 12 Bit den Wert 1 hat, springt LoRa um 30,52 Hz, hat es den Wert 2, ist die Sprungweite doppelt groß, und das geht so weiter bis zum Wert 4096. In Wirklichkeit sind die Dinge etwas komplexer, doch an dem vereinfachten Beispiel wird das Prinzip deutlich. Nebenbei bemerkt: Im Netz wurde eine umfassende Analyse von LoRa [3] veröffentlicht.

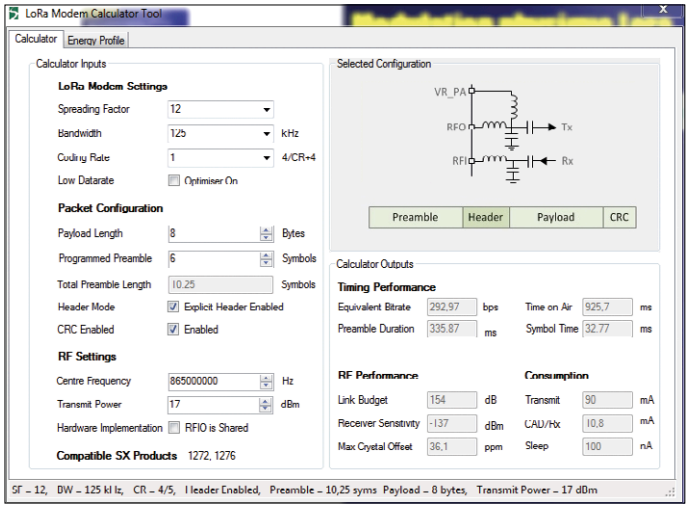


Bild 6. Das Programm *LoRa Modem Calculator* von Semtech ist ein nützliches und vielseitiges Tool.

Software-Tool

Das Berechnen der effektiven Geschwindigkeit einer LoRa-Übertragungsstrecke als Funktion der diversen Parameter ist nicht unkompliziert. Semtech stellt jedoch ein Tool zur Verfügung [4], das auf der Windows-Plattform läuft und diese Arbeit übernimmt. In **Bild 6** ist das vom Tool gelieferte Ergebnis wiedergegeben, wobei die Vorgaben von dem oben betrachteten Beispiel übernommen wurden (BW = 125 kHz, SF = 12, Korrekturcode 5/4). Das Tool bestätigt die effektive Übertragungsgeschwindigkeit 292 bps, wie sie bereits abgeschätzt wurde. Das Programm ist auch in der Lage, die Zeit zu berechnen, die zum Senden eines Frames mit einer gewünschten Anzahl nutzbarer Bytes und den begleitenden Bytes benötigt wird. Um eine Größenordnung zu nennen: Die Übertragungszeit für acht nutzbare Bytes beträgt mit den vorstehenden Parametern 925 ms. Damit stößt sie an eine Beschränkung, der alle Übertragungsverfahren mit niedrigen Geschwindigkeiten wie LoRa unterliegen: In Europa sind in den öffentlichen Frequenzbändern die Übertragungszeiten begrenzt, um auch Mitbenutzer zum Zug kommen zu lassen. Im Bereich 868,0 MHz bis 868,6 MHz, der von LoRa überwiegend genutzt wird, beträgt die zeitliche Begrenzung 1 %. Auf eine Stunde (= 3600 s) umgerechnet sind dies nur 36 s. Aus diesem Grund ist es nicht möglich, in einer Stunde mehr als etwa 40 Nachrichten bestehend aus 8 Byte zu übertragen. Umfangreiche Dateien scheiden für die Übertragung aus, LPWA-Verfahren haben ausschließlich kurze Daten-Telegramme im Fokus. Abhilfe könnten höhere Übertragungsgeschwindigkeiten schaffen, doch dann würden wieder die Reichweiten sinken.

Das Windows-Tool von Semtech liefert auch eine Abschätzung der Empfänger-Empfindlichkeit, die bei den gewählten Einstellungen nötig ist. Aus Bild 6 geht hervor, dass der Wert hier bei -137 dBm liegt. Wenn der Sender die Leistung 25 mW (14 dBm) hat, beträgt die Leistungsübertragungsbilanz (*Link Budget*) 151 dB, was gleichbedeutend mit einer Reichweite im offenem Feld von mehr als 900 km ist. Wegen der Hindernisse

und Reflexionen auf der Übertragungsstrecke ist dies natürlich nur ein theoretischer Wert. Doch mit smarten Systemen und günstigen Aufstellorten der Antennen lassen sich Distanzen von mehreren Dutzend Kilometern sicher überbrücken.

Und die Hardware?

Wie können Sie LoRa zusammen mit Ihrem Projekt einsetzen? Zunächst können Sie nur die physische LoRa-Schicht nutzen und eine unmittelbare Verbindung zwischen den Systemkomponenten installieren. Oder Sie warten auf die Beschreibung von LoRaWAN, die im nächsten Beitrag dieser Reihe folgt. Anschließend müssen Sie entscheiden, ob Sie einen Chip oder ein fertiges Modul bevorzugen. Die Palette der Chips ist begrenzt, denn der einzige Hersteller von Chips, die LoRa unterstützen, ist Semtech. Daran kann sich jedoch im Verlauf des Jahres 2017 etwas ändern, denn STM und Microchip verhandeln mit Semtech über die Vergabe von Lizenzen. Gegenwärtig empfehlen wir Ihnen, die Website von Semtech zu besuchen und die Dokumentation des SX1276 herunterzuladen [5]. Der Typ SX1276 ist im Frequenzband 137...1040 MHz funktionsfähig, während der kostengünstigere SX1272 nur im 868-MHz-Band arbeitet.

Beide Versionen kommen mit wenigen externen Komponenten aus: Ein Quarz, einige passive Bauelemente, eine Antennenbuchse sowie ein Mikrocontroller, der die Steuerung übernimmt. Ein 32-bit-Typ, zum Beispiel ein Cortex M0, genügt vollkommen. Was die Software betrifft, ist für den Anfang genügend Open-Source-Material auf Github vorhanden (www.github.com, nach „SX1276“ suchen).

Die zweite, unkompliziertere Option ist der Einsatz eines gebrauchsfertigen Moduls. Das Modul enthält einen Chip, der LoRa unterstützt, sowie einen vorprogrammierten Mikrocontroller, der den LoRa-Chip steuert. Schon jetzt ist die Auswahl nicht unbeträchtlich, ein Beispiel ist der Typ RN2483 von Microchip (**Bild 7**) mit einem Preis unter 15 €. Das Modul muss lediglich über eine UART-Schnittstelle mit dem Host-Mikrocontroller verbunden werden. Nach wenigen ASCII-Kommandos ist LoRa an den Host angebunden; was dann noch fehlt, ist die Antenne. Arduino-Enthusiasten können auf ein Shield mit dem SX1276 und anderes notwendiges Zubehör zurückgreifen, das mit der Mbed-Entwicklungsumgebung des Chip-Spezialisten ARM kompatibel ist (SX1276MB1xAS, siehe [7]). Im Elektor-Shop ist ein HAT für den Raspberry Pi erhältlich [8]. Dem Projektstart steht eigentlich nichts im Weg!

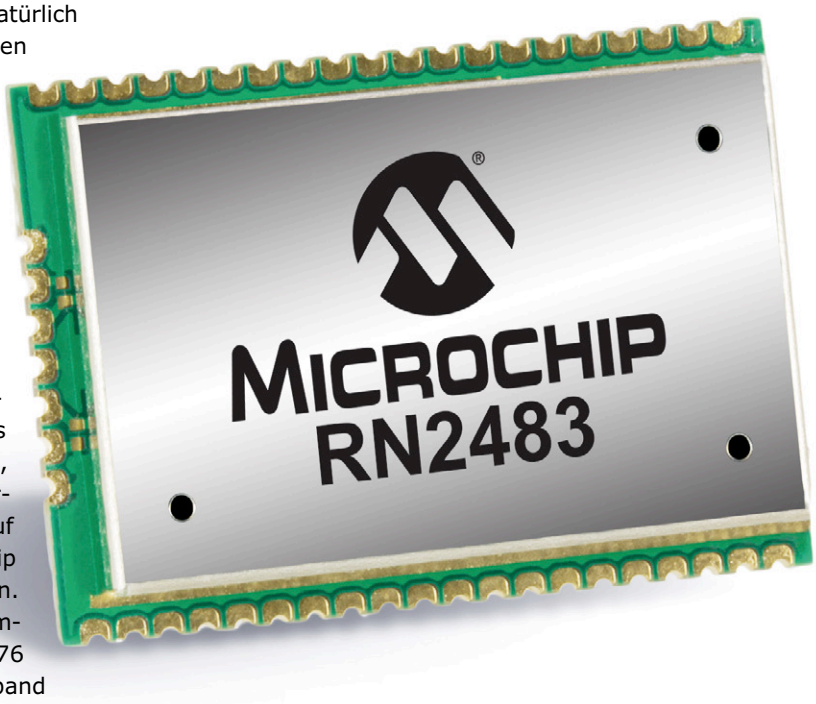


Bild 7. Beispiel für ein sofort einsatzfähiges LoRa-Modul.

Resümee

LPWA-Netzwerke, insbesondere LoRa, machen kostengünstige Daten-Funkverbindungen über weite Entfernungen möglich. Die gravierendste Einschränkung ist jedoch, dass die übertragbaren Datenmengen ebenso wie die Übertragungsgeschwindigkeiten extrem niedrig liegen. Trotzdem existiert eine Fülle potentieller Anwendungen.

Thema des nächsten Beitrags dieser Reihe wird LoRaWAN sein, ein Protokoll, das von einem Konsortium gleichen Namens entwickelt wurde und sich auf LoRa stützt. Die Tatsache, dass sich in der LoRa-Allianz mehr als 200 nicht nur kleine Unternehmen zusammen gefunden haben, verspricht einiges für die Zukunft. Zur LoRa-Allianz gehören Namen wie Cisco, IBM, ARM, Orange, Bouygues Telecom, KPN, Proximus, STMicroelectronics und Microchip. ◀

(160236)gd

Dieser Beitrag ist auch in Circuit Cellar Nr. 313 erschienen.

Weblinks und Quellen

[1] *Rauschen und Empfindlichkeit bei Empfängern,* Elektor März/April 2017: www.elektormagazine.de/160307

[2] AN1200.22, *LoRa Modulation Basics*: www.semtech.com/images/datasheet/an1200.22.pdf

[3] *Reversing Lora*, Mark Knight / Bastille networks: https://archive.org/details/ReversingLora

[4] Tool LoRa Calculator – *Fast evaluation of link budget, time on air and energy consumption*: www.semtech.com/wireless-rf/rf-transceivers/sx1272

[5] LoRa-Transceiver SX1276: www.semtech.com/wireless-rf/rf-transceivers/sx1276

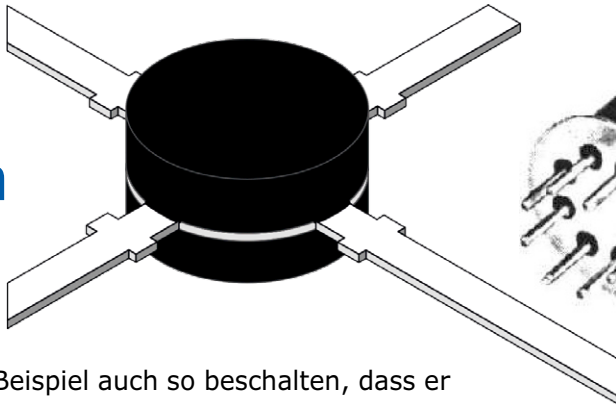
[6] LoRa-Drahtlos-Modul RN2483: www.microchip.com/RN2483

[7] LoRa-Shield SX1276MB1xAS: https://developer.mbed.org/components/SX1276MB1xAS

[8] LoRa-HAT für RPi: www.elektor.de/dragino-lora-gps-hat-for-raspberry-pi

Picowatt-Sender

UKW-Sender mit BF961 als Fetron



Von **Hans-Norbert Gerbig** (D)

Ein BF961 ist ein „normaler“ Dual-Gate-MOSFET. Den kann man zum Beispiel auch so beschalten, dass er als Fetron fungiert. Und mit diesem Fetron lässt sich ein simpler FM-UKW-Sender realisieren, der nur wenig Strom braucht und dessen geringe Ausgangsleistung sehr gut zum Test von UKW-Radios etc. taugt. Wenn Sie jetzt über den Begriff Fetron gestolpert sind, sollten Sie weiterlesen...

Ein kleiner FM-UKW-Sender sollte in keiner Werkstatt und keinem Labor fehlen. So ein Sender eignet sich sehr gut zum Test diverser Empfänger – er sollte aber nur eine sehr geringe Leistung abstrahlen, damit praktisch auf Armlänge schon nichts zu messen ist, um Störungen offizieller Funkdienste und Konflikte mit Verordnungen und gesetzlichen Regelwerken zu vermeiden. Dies ist mit der folgenden Schaltung sehr einfach zu realisieren.

Fetron

Die Firma Teledyne Semiconductor hat vor langer Zeit im Jahre 1972 – aus heutiger Perspektive kurz nach Erfindung des Transistors - einen speziellen Halbleiter vorgestellt, der als direkter Ersatz von Röhrenpentoden gedacht war. Die luftleeren Glaskolben mit Glühdraht drin waren ja Anfang der 1970er Jahre akut vom Aussterben bedroht.

Der Fetron sollte zumindest Pentoden ersetzen können und somit dem einen oder anderen tollen Gerät oder bewährten Schaltungen ein längeres Leben bescheren.

Bild 1 zeigt die revolutionäre Erfindung in voller Pracht: Durch Kaskadenschaltung zweier n-Kanal-Hochvolt-FETs wurde ein dreipoliger Halbleiter konstruiert, der einen ähnlichen Kennlinienverlauf wie eine Pentode aufweist. Dank hoher Maximalspannung von 350 V, hoher Maximalfrequenz und auch sonst ähnlicher Eigenschaften lassen sich damit also tatsächlich direkt Röhren ersetzen und dabei auch noch die Heizung sparen. Unter [1] ist ein originales Angebot des Distributors Omni Ray zu finden, der dieses phänomenale Halbleiterbauteil zum ebenfalls sagenhaften Preis von 46 Schweizer Franken – pro Stück! – angeboten hat.

UKW-Sender

Hohe Frequenz und Pentodenverhalten – damit lassen sich doch prima HF-Schaltungen aufbauen, dachte sich der Autor. Doch heute einen (funktionierenden) Fetron zu bekommen, dürfte außergewöhnliches Glück erfordern. Also musste ein Ersatz des Röhrenersatzes her. Und tatsächlich geht das: Man kann einen modern(er)en Dual-Gate-MOSFET nämlich durchaus

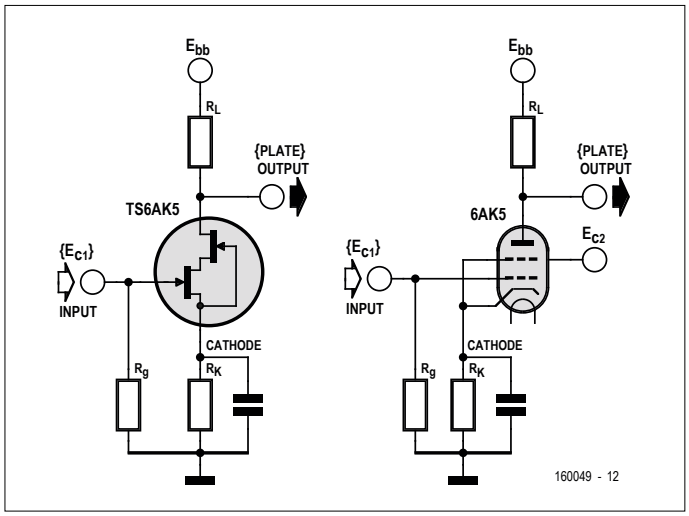


Bild 1. Röhrenpentode 6AK5 (rechts) und Ersatz durch ein Fetron mit dem bezeichnenden Namen TS6AK5 von Teledyne (links).

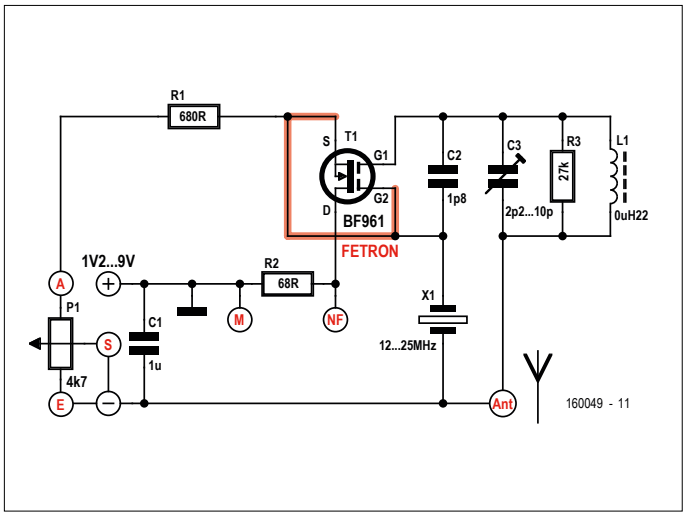
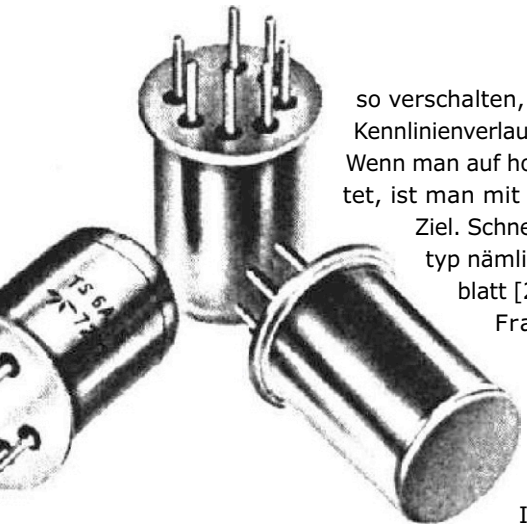


Bild 2. Schaltung des Picowatt-Senders mit Ersatz-Fetron in Source-Schaltung.



so verschalten, dass er einen ähnlichen Kennlinienverlauf wie ein Fetron aufweist. Wenn man auf hohe Spannungen verzichtet, ist man mit einem BF961 schon am Ziel. Schnell ist dieser Ersatzersatztyp nämlich auch, wie sein Datenblatt [2] belegt. Die spannende Frage ist: Wie wird aus einem Dual-Gate-MOSFET ein Fetron-ähnliches Bauteil?

Die einfache Antwort: Indem man sein Gate2 mit der Source verbindet. **Bild 2** zeigt, wie aus dem Ersatz-Fetron mit Hilfe einiger Zusatzbauteile ein frequenzmodulierbarer HF-Sender sehr kleiner Leistung wird, der dank seiner inhärenten Hochohmigkeit mit erstaunlich wenig Betriebsstrom auskommt. Der Pseudo-Fetron ist hier in Source-Schaltung eingesetzt. Das modulierende Signal wird niederohmig am Eingang NF gegen Masse (Punkt M) eingespeist. Es variiert die Versorgungsspannung. Der Witz ist nun, dass der Oszillator dabei auf seiner Eigenfrequenz mit der gleichen Amplitude konstant weiter schwingt. Der Mitnahmebereich der Eigenfrequenz entspricht genau der steuernden NF, es wird aber die links und rechts vom Mitnahmebereich störende Amplitudenmodulation quasi abgeschnitten. Das Resultat ist die gewünschte Frequenzmodulation!

Zur Funktion: Die Rückkopplung des Oszillators erfolgt vom Source/Gate2-Anschluss auf den in Serie mit dem Kondensator C2 liegenden Quarz, parallel zum Schwingkreis. Mit dem Trimmer C3 stellt man den Oszillator auf Resonanz mit einem Vielfachen der Quarzfrequenz. Die modulierende Niederfrequenz muss niederohmig eingekoppelt werden. Hierzu eignet sich ein kleiner NF-Verstärker kleiner Leistung oder ein Kopfhörer-/Lautsprecher-Ausgang eines Radios etc. C1 dient zur Reduktion von Brummeinstreuungen.
Erstaunlich sind der geringe Stromverbrauch und der große Betriebsspannungsbereich von 1,2...9 V. Der Oszillator schwingt schon ab 0,3 mA. Eine sichere Standardeinstellung für den gesamten Betriebsspannungsbereich ist schon ab 0,6 mA möglich. Der Testsender ist also prima per Batterie zu versorgen.

Aufbau und Test

Für die recht einfache Schaltung wurde vom Autor auch eine Platine entworfen, deren Layout-Dateien von der Elektor-Webseite zu diesem Artikel [3] kostenlos downloadbar sind. Dass es sich dabei um Dateien des nicht ganz so verbreiteten Programms Sprint-Layout handelt, ist kein großes Problem, denn die Schaltung ist so simpel, dass man sie „trotz HF“ bequem auf einer Streifenleiter-Experimentierplatine unterbringen kann. Und hierfür gibt es mit **Bild 3** sogar eine passende Vorlage. Die paar Bauteile sind schnell bestückt. Anstelle eines richtigen Potis an den drei Punkten A, S und E kann man die Platine natürlich auch links etwas länger lassen und ein Trimmpoti fest auflöten. Hierzu eignet sich die größere Trimmpoti-Bauform. Man muss lediglich beim MOSFET etwas aufpassen, da diese Bauteile bekanntlich empfindlich auf elektrostatische Aufladungen reagieren. Also beim Hantieren erden...
Nach Bestückung und Überprüfung kann man eine geeignete Spannungsquelle und ein Audiosignal anschließen. P1 wird auf die empfohlenen 0,6 mA Stromaufnahme eingestellt. Anschließend stimmt man ein neben dem Sender stehendes UKW-Radio durch. Sollte der Sender nicht empfangbar sein, verdreht man den Trimmer C3 etwas und probiert es nochmals. Hat man seinen Sender gefunden, kann man die Amplitude des Audiosignals so einstellen, dass es im empfangenden Radio nicht verzerrt klingt. Fertig.

Noch ein Hinweis: Mit steigender Spannung erhöht sich die HF-Leistung. Für ein direkt neben dem Sender stehendes Radio ist es nicht erforderlich, eine Antenne anzuschließen. Ein paar Picowatt werden auch so abgestrahlt. ⏮

(160049)

Weblinks

[1] Fetron: http://goo.gl/RrIOW8
[2] BF961: http://goo.gl/xNha5N
[3] www.elektormagazine.de/160049

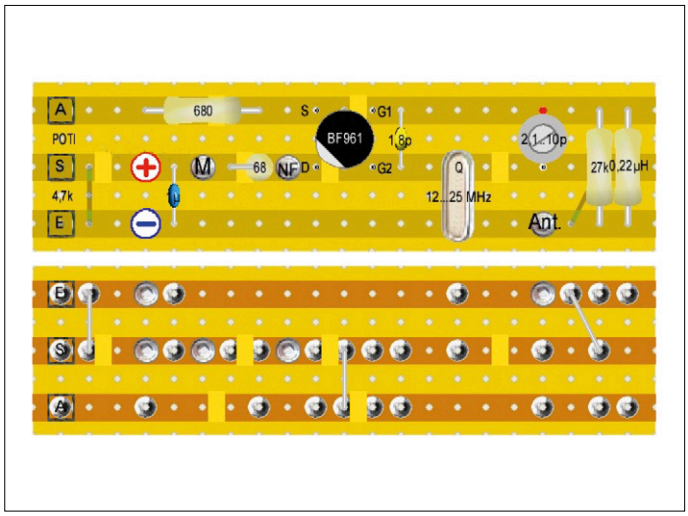


Bild 3. Ein Stückchen Lochrasterplatine mit drei Leiterbahnen genügt zum Aufbau des Senders.

STÜCKLISTE

Widerstände:
Standard: Kohleschicht, 5 %
R1 = 680 Ω
R2 = 68 Ω
R3 = 27 k
P1 = 4k7, Poti oder Trimmpoti*

Kondensatoren:
C1 = 1 µ, keramisch oder Folie*
C2 = 1p8, keramisch
C3 = 10 p, Trimmer*

Spule:
L1 = 0,22 µ, Festinduktivität

Halbleiter:
T1 = BF961, z.B. Vishay

Außerdem:
X1 = 12...25 MHz, Oberwellenquarz, Serienresonanz
Antenne*
Platine oder Experimentierplatine 57,5 x 17,5 mm*

* siehe Text

UV-Belichter mit PIC-Timer

Matrix aus 54 UV-LEDs

Von **Steve Griffin** (UK)

UV-LEDs zur Belichtung fotoempfindlicher Platinen einzusetzen ist zwar keine neue Idee. Doch diese Schaltung mit passendem Timer vereinfacht den Bau einer nach unten strahlenden Belichtungseinheit, die in ein handelsübliches Gehäuse eingebaut werden kann.

Dieser Artikel ist ein aufbereitetes Konzentrat und ein Zeiger auf eine umfangreichere Beschreibung, die Elektor-Mitglieder auf elektormagazine.de/labs lesen können. Während hier genug Raum für die Beschreibung der Schaltung und einiger anderer Aspekte ist, dreht sich der Artikel auf der Webseite [1] um den Aufbau und einige Optionen sowie die PIC-Software und die Platine.

Was ist drin?

Die Belichter-Elektronik besteht aus zwei Platinen: eine für die reine UV-Belichtung und die andere für den Mikrocontroller-basierten Timer. Die UV-Platine ist mit 54 UV-LEDs bestückt, die in einer Matrix angeordnet sind. Die Timer-Platine beherbergt einen einfachen PIC-Mikrocontroller und wird später auf die UV-Platine montiert. Die Programmierung macht aus der MCU einen einstellbaren Timer mit Siebensegment-Display und zwei Timer-Ausgängen (einer für einseitige Platinen und der zweite für eine Erweiterung für doppelseitige Platinen). Außerdem ist noch ein Tonausgang vorgesehen, der das Ende der Belichtungszeit akustisch signalisiert. Typischerweise muss man hier mit etwa 2,5 Minuten rechnen.

Timer-Platine

Die Timer-Schaltung (**Bild 1**) wird auf einer doppelseitigen Platine realisiert, die direkt über der umgedrehten UV-Platine montiert wird. Dadurch entsteht eine sehr kompakte Einheit. Mit J7 gibt es auch einen ICSP-Anschluss, an den sich ein In-Circuit-Programmer anschließen lässt. Versorgt dieser die Elektronik beim Programmieren, muss währenddessen die Stromversorgung des Belichters inaktiv sein.

Der PIC 16F877A ist gut erhältlich und bietet viele leicht zugängliche Ports. Er wird von einem 5-V-Linearregler versorgt, der seinerseits vom 24-V-Steckernetzteil gespeist wird. Dieses Netzteil versorgt dank seiner Belastbarkeit mit 1 A direkt die UV-Platine(n). Die LED D2 leuchtet, wenn die Stromversorgung steht. Der MCU-Takt wird von einem 20-MHz-Quarz generiert.

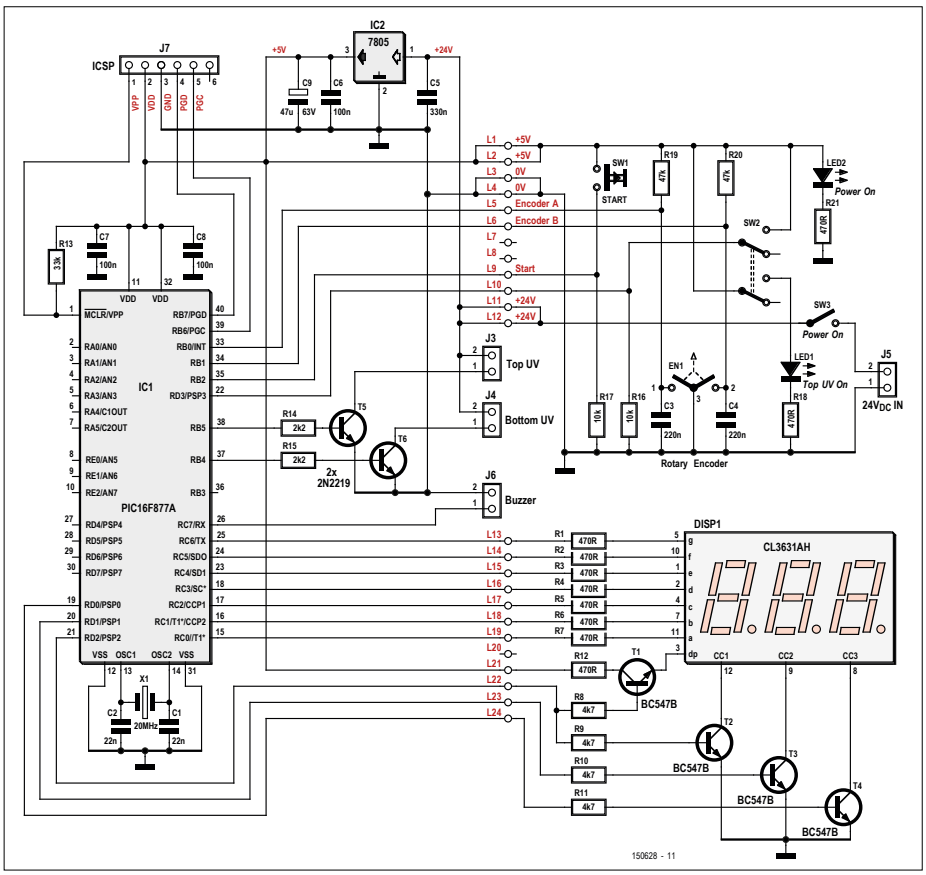


Bild 1. Schaltung der Timer-Platine samt Bedienelementen und dreistelligem Siebensegment-Display.

Die Ports C und D dienen als Ausgänge und steuern das dreistellige Siebensegment-Display. Die Stellen werden von den drei Transistoren T2...T4 angesteuert. Der durch T1 angesteuerte Dezimalpunkt trennt die Minuten von der Sekundenanzeige. Von den übrigen Leitungen wird C7 zur Ansteuerung eines Buzzers und B5 zum Einschalten der optionalen zweiten UV-Platine genutzt. Letzteres wird durch die LED D1 signalisiert.
Alle anderen Funktionen liegen an Port B. Zwei Leitungen steuern zwei ungeschützte Transistoren, die mit ihren Open-Collector-Ausgängen die UV-Platinen schalten. Die anderen Leitungen dienen als Eingänge. Mit ihnen wird der Start-Taster und die Betätigung des Dreh-Encoders zur Einstellung der Belichtungszeit abgefragt.
Die meisten Funktionen liegen auf der Hand. Lediglich die Behandlung des Encoders ist etwas komplexer. Der verwendete Dreh-Encoder hat 16 Stellungen pro Umdrehung und kann über 360° hinaus in beide Richtungen gedreht werden. Jeder Schritt in Uhrzeigerrichtung erhöht die Zeit um 15 s. Umgekehrt reduziert jeder Schritt zurück die Zeit um ebenfalls 15 s. Der einstellbare Zeitbereich geht von 15 s bis zu 9 min und 45 s. Da eine Drehung des Encoders Schaltvorgänge auslöst, braucht es auch eine zuverlässige Entprellung für die Eingänge des Mikrocontrollers. Diese wird durch die RC-Glieder R19/C3 und R20/C4 mit Zeitkonstanten von etwa 10 ms vorgenommen.
Der Betrieb ist sehr simpel. Nach dem Einschalten wählt man mit EN1 die gewünschte Zeit und betätigt dann den Start-Knopf. Dann leuchten die UV-LEDs (wenn der Deckel geschlossen ist). Wenn der Countdown auf dem Display die Null erreicht hat, werden die UV-LEDs wieder abgeschaltet und der Buzzer macht fünf Töne.

UV-Platine

Die UV-Platine hat die Abmessungen 160 x 100 mm und ist einseitig. Man sieht gleich, dass der Belichter für das klassische Eurokarten-Format vorgesehen ist. Die „Schaltung“ (**Bild 2**) besteht aus neun Spalten mit Reihen zu je sechs UV-LEDs. An einer UV-LED fallen beim Nennstrom von 20 mA etwa 3,4 V ab. Das verwendete Steckernetzteil liefert unter Last genau 23,7 V. Der nötige Vorwiderstand pro Reihe errechnet sich daher zu:

$$23{,}7\ V - (6 \times 3{,}4\ V) = 3{,}1\ V$$

Da $R = U / I$ ergibt sich bei I = 20 mA:

$$R = 3{,}1\ V / 20\ mA = 165\ \Omega$$

Ein E12-Wert von 150 Ω eignet sich daher prima zur Strombegrenzung.
Die UV-LEDs sind so angeordnet, dass sie zu allen Nachbarn ungefähr gleiche Abstände aufweisen, was eine gleichmäßige Belichtung garantiert. Die Platine ist außerdem so ausgelegt, das zwei davon nebeneinander zu einer größeren Fläche montiert und parallelgeschaltet werden können, was mit 160 x 200 mm das doppelte Eurokarten-Format ergibt. Die beiden Transistoren T5 und T6 sind durchaus dazu in der Lage, gleich vier parallel geschaltete UV-Platinen anzusteuern. Der ideale Abstand zwischen den UV-LEDs und der zu belichtenden beschichteten Platine liegt bei 100...150 mm.

Flexibilität

Timer- und UV-LED-Platine sind für eine Sandwich-Montage vorbereitet (**Bild 3**). Diese Konstruktion wird dann mit vier Schrauben und Abstandshaltern am Deckel des Gehäuses fixiert. Je nach Ausführung ergeben sich unterschiedliche Stromverbräuche: Bei einer einseitigen Version mit 160 × 100 mm ergeben sich 180 mA und bei der doppelten Fläche von 160 × 200 mm entsprechend 360 mA. Macht man dies doppelseitig, muss man mit 360 bzw. 720 mA rechnen. Der Belichter muss mit stabilisierten 24 V versorgt werden. Alle denkbaren Optionen werden auf Elektormagazine.de/labs [1] detailliert beschrieben. Dort gibt es auch noch eine B-Variante der UV-Platine mit zusätzlichen 18 weißen Weitwinkel-LEDs. Mit Hilfe eines Schalters kann man die Anordnung dann auch als sichere Licht-Box verwenden. Die verwendeten LEDs haben einen sehr großen Abstrahlwinkel, weswegen weniger Exemplare für eine gleichmäßige Abdeckung erforderlich sind. Außerdem wird noch ein einstellbarer Widerstand eingeschleift, mit dem die gewünschte Helligkeit eingestellt werden kann.
Diese Schaltung enthält auch noch Sicherheitsmaßnahmen, die verhindern, dass jemand aus Versehen direkt in die UV-LEDs schaut. Die Gesamtstromaufnahme dieser Variante beträgt rund 420 mA. ◀

(150628)

Weblink

[1] Artikel auf /labs: www.elektormagazine.de/labs/uv-exposure-unit-with-built-in-pic-based-timer

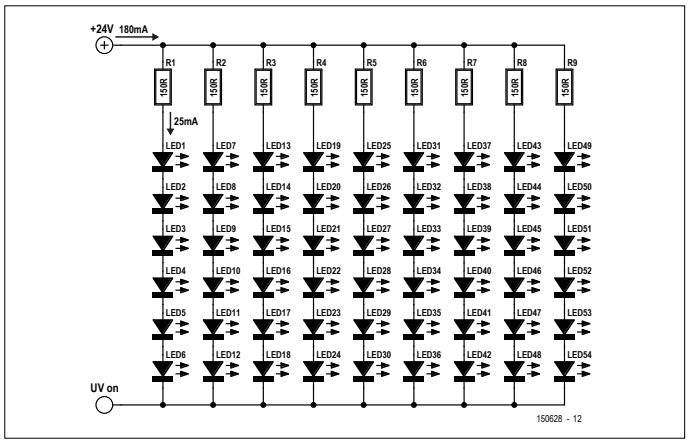


Bild 2. Schaltung der UV-Platine.

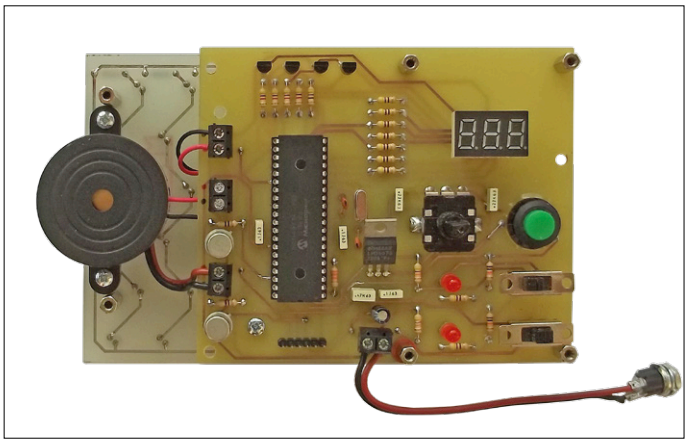
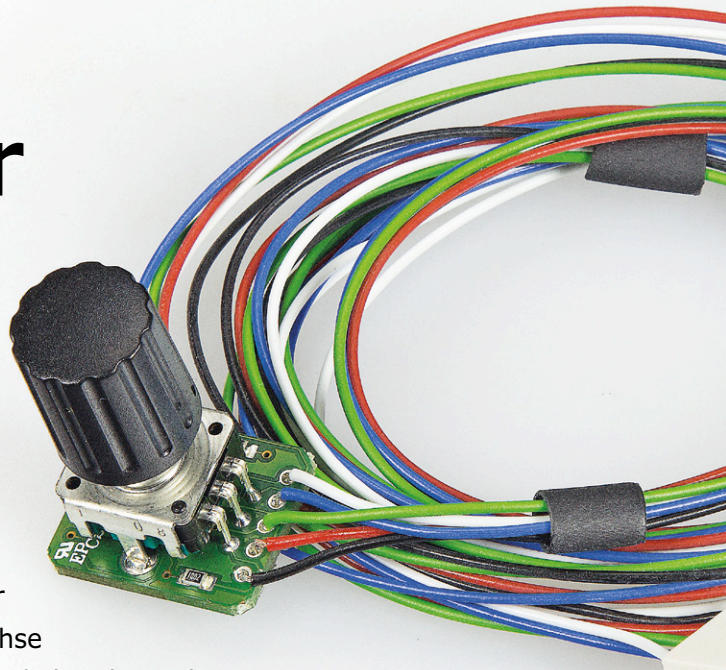


Bild 3. Montage von Timer- und UV-Platine. Der Buzzer gehört zwar elektrisch zur Timer-Platine, wird aber auf der Rückseite der UV-Platine bestückt.

Drehimpulsgeber anschließen...

...ohne Mikrocontroller!



Von **Mike Mittelhäuser** (D)

Inkrementelle Drehimpulsgeber sind elektromechanische oder optoelektronische Komponenten, die beim Drehen an ihrer Achse pulsförmige Signale abgeben. Die Signale hängen vom Drehwinkel und von der Drehrichtung ab. Eingesetzt werden Drehimpulsgeber dort, wo das präzise Erfassen digitaler oder analoger Größen notwendig ist. Üblicherweise interpretiert ein Mikrocontroller die Signale, bevor sie weiter verarbeitet werden können. Dass es auch ohne Mikrocontroller geht, beweist dieses Leserprojekt.

Ein Mikrocontroller mit zugehöriger Firmware, nur um den Drehwinkel und die Drehrichtung zu bestimmen, ist ein vergleichsweise hoher Aufwand. Ein Projekt des Autors erforderte den flexiblen Einsatz von einigen Dutzend Drehimpulsgebern, sowohl für digitale als auch analoge Funktionen. Dort lag es nahe, nach einer ökonomischeren Lösung zu suchen. Die Schaltung, so war das Ziel, muss mit wenigen Standard-Logikbausteinen auskommen und auf einer kleinen Platine in der Größe des Drehimpulsgebers Platz finden. An den Ausgängen müssen beim Rechts- und Linksdrehen separate pulsförmige Signale erscheinen, bei schnellen Drehungen dürfen sich die Impulse nicht überlappen.

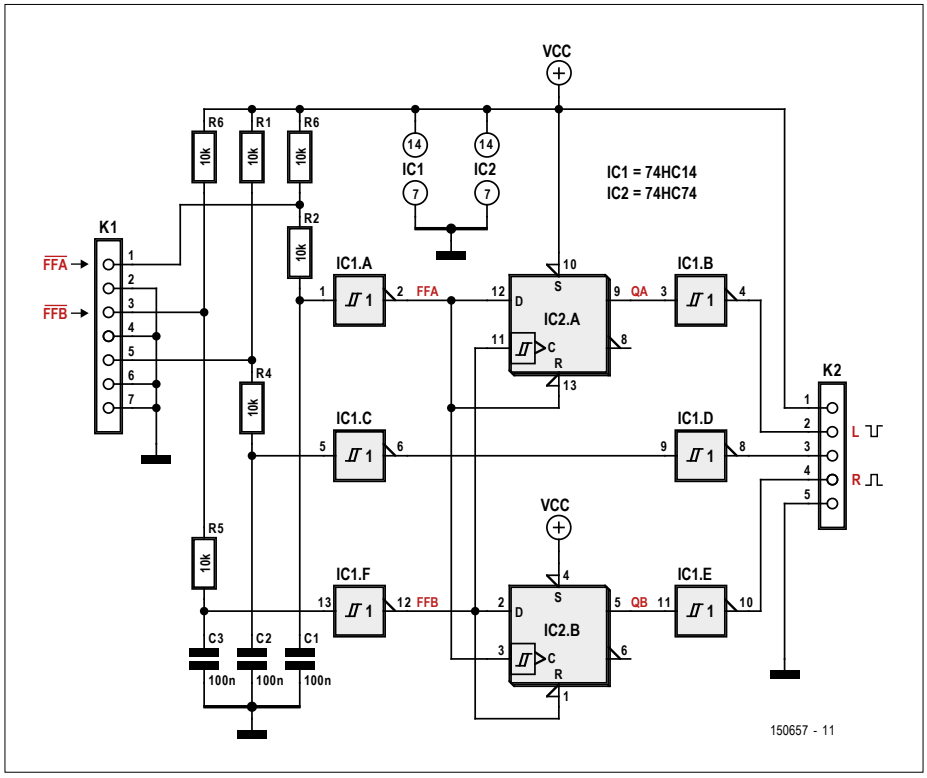


Bild 1. Die Signale des Drehimpulsgebers werden mit zwei Logik-ICs aus der 74HC-Reihe dekodiert.

Zwei Logik-ICs

Die Schaltung des Autors ist für Drehimpulsgeber mit integriertem Drucktaster und ohne Drucktaster geeignet. Anschließbar sind die meisten auf dem Markt befindlichen Typen, darunter auch die Produkte des Herstellers Alps.

Wie **Bild 1** zeigt, besteht die Schaltung aus zwei positiv flanken-getriggerten D-Flipflops mit Setz- und Rücksetz-Eingang in CMOS (IC2.A und IC2.B, 74HC74) sowie sechs Invertern mit Schmitt-Trigger-Eingängen (IC1.A…F, 74HC14), ebenfalls in CMOS. Die Signale des Drehimpulsgebers einschließlich Drucktaster sind aktiv low, jeder Eingang wird deshalb über zwei in Reihe liegende 10-kΩ-Widerstände auf die Betriebsspannung gezo-

gen. Die Kondensatoren C1…C3 entkoppeln die Eingangssignale. Die Schaltung liefert prellfreie, low-aktive pulsförmige Signale, die negativ gerichteten Impulse sind 7 ms, die positiv gerichteten Impulse 14 ms lang.

Die Schmitt-Trigger-Inverter an den Eingängen invertieren die vom Drehimpulsgeber kommenden, negativ gerichteten Impulse und erhöhen die Flankensteilheit, so dass die nachfolgenden D-Flipflops störungsfrei getriggert werden können. Um ein Überlaufen zu vermeiden, muss die Zeit zwischen aufeinander folgenden Impulsen mindestens 10,5 ms lang sein.

Das Ausgangssignal des Inverters IC1.A liegt am D-Eingang und am Reset-Eingang des ersten Flipflops IC2.A. Der Set-Eingang hat keine Funktion, er ist deshalb mit der Betriebsspannung verbunden. Außerdem liegt das Signal am Takt-Eingang des zweiten Flipflops IC2.B. Das Ausgangssignal des Inverters IC1.F gelangt in gleicher Weise zu den Flipflops, nur mit dem Unterschied, dass IC2.A und IB2.B vertauscht sind.

Solange der Drehimpulsgeber in Ruhe ist, liegen die Ausgänge beider Flipflops auf Low, die invertierten Signale an K2 sind high. Wenn der Drehimpulsgeber einen Schritt ausführt, erscheinen auf seinen Leitungen zwei gegeneinander phasenverschobene Impulse. Die Art der Verschiebung hängt von der Drehrichtung ab.

Drehbewegungen

Beim Drehen nach links erscheint zuerst ein Impuls ($\overline{FFA}$) auf Leitung 1 von K1 (FFA in **Bild 2**), nach einer viertel Periode gefolgt von einem Impuls auf Leitung 3 ($\overline{FFB}$). Der erste, low-aktive Impuls ($\overline{FFA}$) wird von IC1.A invertiert, er gibt den Reset von IC2.A frei. Da das Taktsignal auf Leitung 3 von K1 noch nicht eingetroffen ist, bleibt der Ausgang von IC2.A vorläufig low, der zugehörige Ausgang auf K2 ist high. Flipflop IC2.B wird jetzt zwar getaktet, sein Ausgangssignal bleibt jedoch unverändert, weil sein D-Eingang low war.

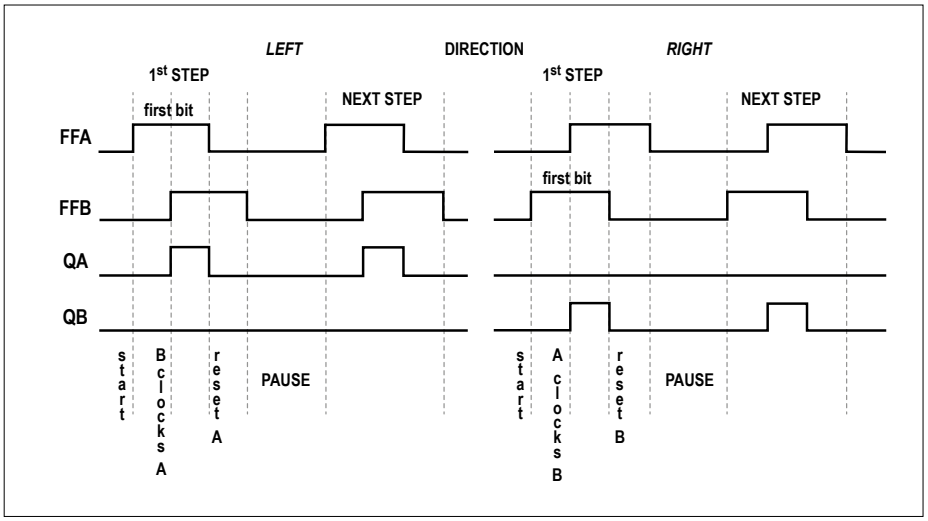
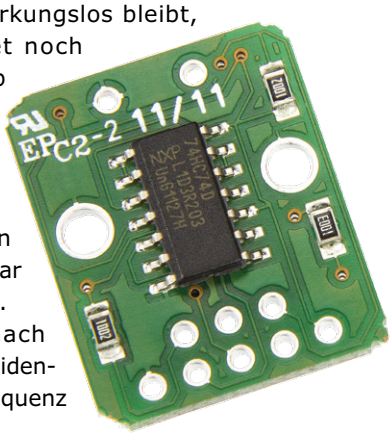
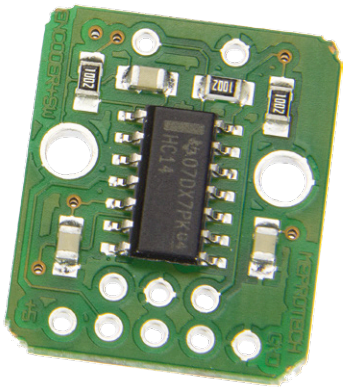


Bild 2. Das Diagramm macht deutlich, wie die Schaltung die zwei in der Phase verschobenen Pulse in zwei separate Puls-Signale umsetzt.

Wenn nun auch auf Leitung 3 von K1 ein low-aktiver Impuls erscheint (Zeit *B clock A* in Bild 2), geht der Ausgang von IC1.F auf High, so dass IC2.A das Signal an seinem D-Eingang auf seinen Ausgang legt. Leitung 2 von K2 wird folglich low zum Zeichen dafür, dass der Drehimpulsgeber nach links dreht. Der D-Eingang von IC2.B ist zwar jetzt auf High, doch da das Signal am Takteingang von IC2.B auf High war und nicht von low nach high wechselte, wird das Signal am D-Eingang von IC2.B nicht zum Ausgang des Flipflops weiter gegeben. Leitung 4 von K2 bleibt auf High. Ist der Impuls auf Leitung 1 von K1 beendet (Zeit *reset A* in Bild 2), wird IC2.A rückgesetzt, Leitung 2 von K2 geht wieder auf High. Das Ausgangssignal von IC2.B bleibt unverändert, da der Takteingang nur auf positiv gerichtete Flanken reagiert.

Der letzte Schritt des Zyklus ist das Ende des Impulses auf Leitung 3 von K1 ($\overline{FFB}$). Das Taktsignal von IC2.A geht auf Low, was jedoch wirkungslos bleibt, weil der Reset noch aktiv ist. Flipflop IC2.B wird rückgesetzt, doch auch hier ändert sich nichts, denn sein Ausgang war bereits auf Low. Das Drehen nach rechts löst eine identische Impulssequenz aus, jedoch mit dem Unterschied, dass sich jetzt das Ausgangssignal von IC2.B ändert und das Ausgangssignal von IC2.A auf Low bleibt.





Zum Schluss

Beim Betrachten der Signale fällt auf, dass die Impulslänge am Ausgang die Hälfte der Impulslänge am Eingang beträgt. Der Ausgang des Drucktasters ist so lange auf Low, wie der Taster gedrückt wird. Das Signal wird lediglich von zwei Schmitt-Trigger-Invertern gepuffert.

Der Autor bietet die Schaltung als fertiges Modul an (ohne Drehimpulsgeber), die Kontaktaufnahme ist über [1] möglich. ⏮

(150657)gd

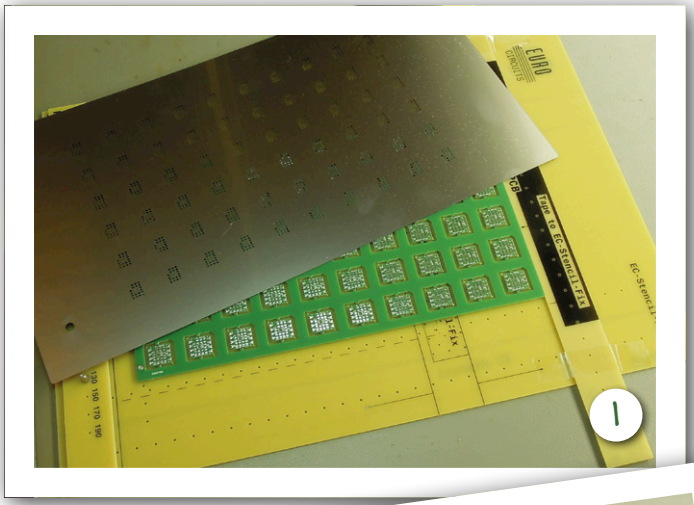
Weblink

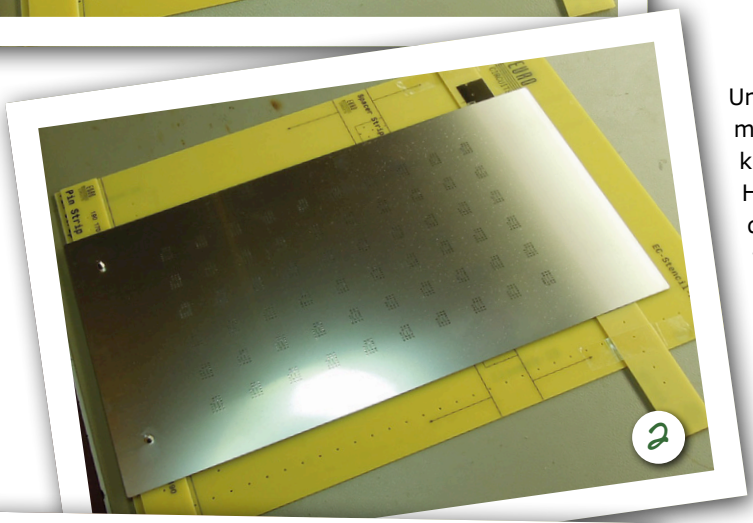
[1] http://mes-engineering.eu; m.mittelhauser@mes-engineering.eu

Arbeiten mit dem eC-stencil-fix

So bauen wir BME280-Sensorplatinen für Sie!

Von **Thijs Beckers**

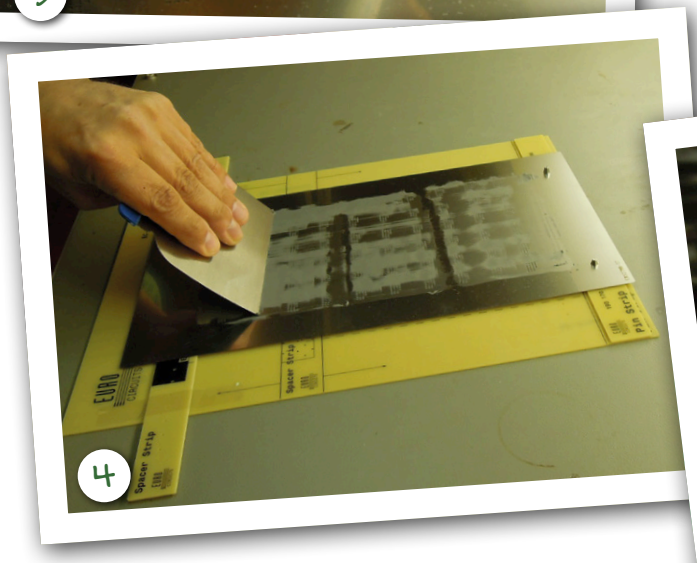


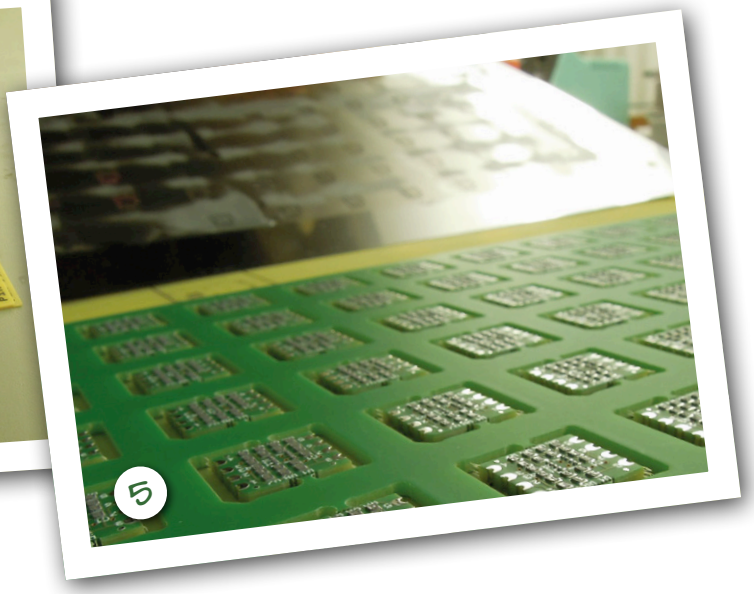


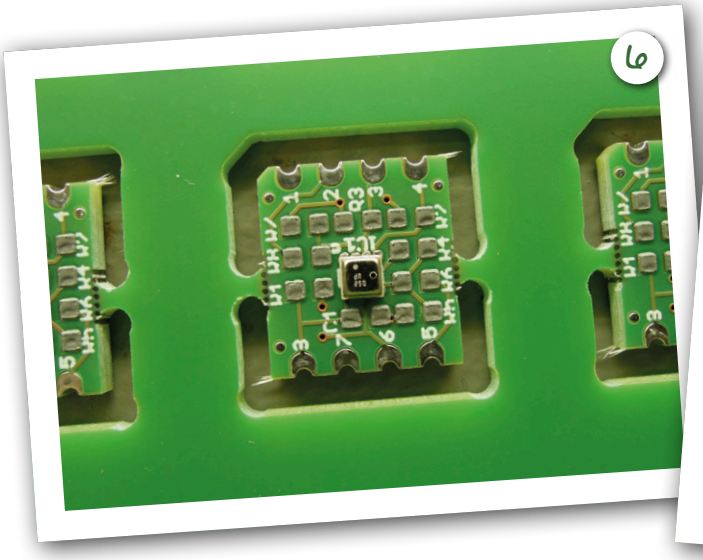
Um den Vorrat unseres Elektor-Shops zu ergänzen, ist es in manchen Fällen schneller/praktischer/einfacher/billiger (oder kurz gesagt besser), wenn wir die Produktion selbst in die Hand nehmen, ganz normal auf dem Labortisch. Dies war der Fall bei zwei kleineren Mengen des Sensorplatinchens für das BME280-Shield [1], das wir Ihnen in der Ausgabe 1-2/2017 vorgestellt haben. Da diese Platine sich dafür ausgezeichnet eignet, haben wir den eC-stencil-fix aus dem Schrank geholt (**Bild 1**).

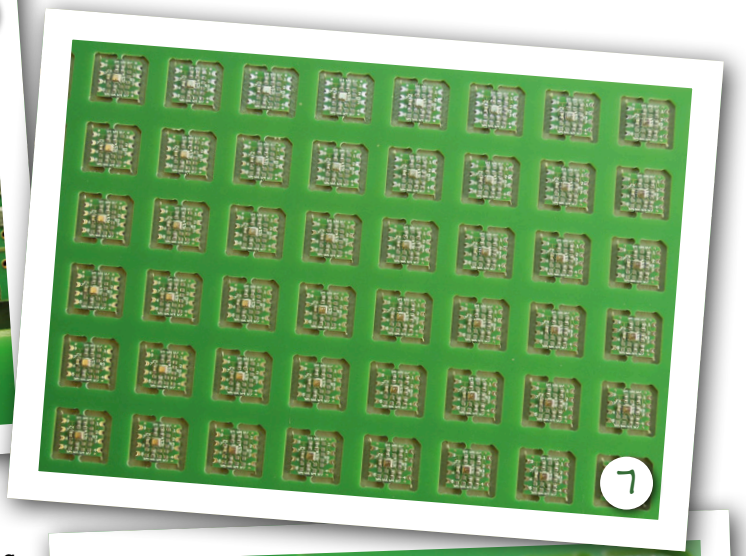
Zunächst werden Platine (oder besser gesagt, die Platinen im „Fertigungsnutzen“) und Stencil korrekt positioniert (**Bild 2**) [2]. Der Nutzen wurde von Eurocircuits [3] während der Produktion am Rand mit kleinen Positionsbohrungen versehen, die exakt an den Stellen sitzen, wo der eC-stencil-fix markiert ist. Die Platine kann so mit zwei „Dornen“ auf einfachste Weise exakt an der richtigen Stelle auf dem eC-stencil-fix angebracht werden.

Zu dem eC-stencil-fix gehören zwei Streifen Platinenmaterial, die für den richtigen Abstand beziehungsweise die korrekte Position der Schablone sorgen. Einer der Platinenstreifen weist Bohrungen auf, in die zwei kleine Positionierbolzen gesteckt werden, die einerseits den Platinenstreifen und andererseits durch ihre Kegelstumpf-Form die Schablone exakt an der richtigen Stelle fixieren. Der zweite Platinenstreifen wird dann an der Platine ausgerichtet und mit Klebeband auf der Unterlage befestigt (und am besten auch die Unterlage am Labortisch). Nun wird ausreichend Lötpaste auf die Schablone (an der







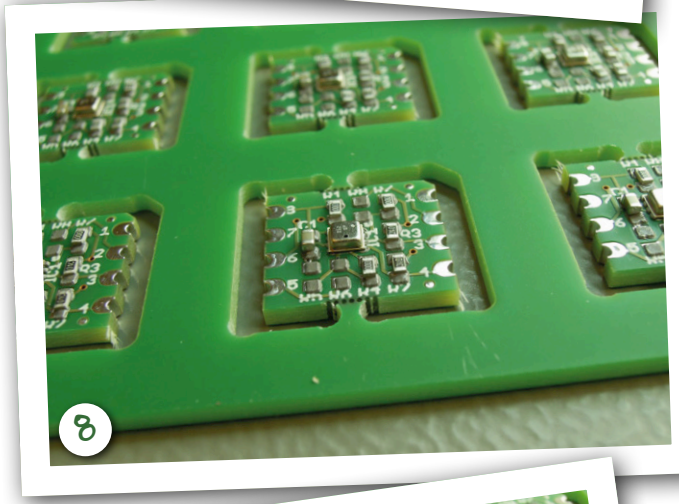


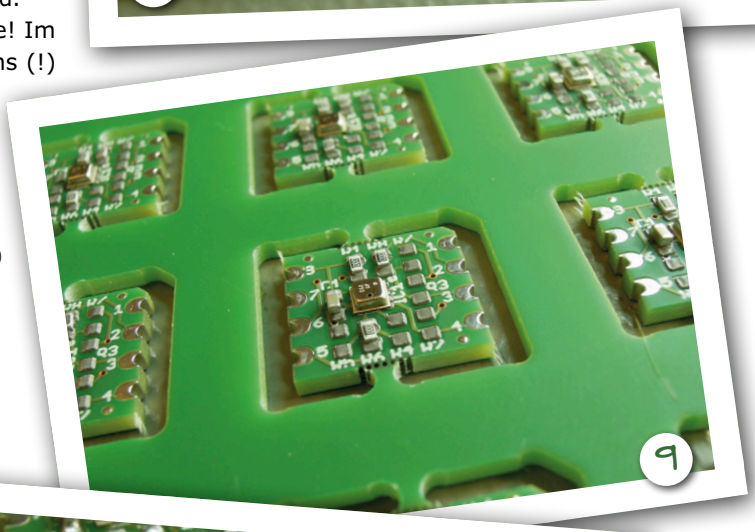
fixierten Seite) aufgetragen (**Bild 3**) und mit einem geeigneten Werkzeug (Spatel, Rakel) über die Schablone gezogen (**Bild 4**). Das Schablonenmaterial ist so dick, dass genau die richtige Menge Lötpaste auf der Platine verbleibt (**Bild 5**). Danach kann die Schablone vorsichtig abgehoben, die Platine entfernt und der Vorgang wiederholt werden, bis alle Nutzen mit Lötpaste versehen sind.

Jetzt war der Nutzen bereit, mit den Bauteilen bestückt zu werden (**Bild 6**). Obwohl er ziemlich klein war, umfasste er doch eine Unzahl der winzigen Sensorplatinchen (**Bild 7**), so dass wir ganz schön zu tun hatten. Wir haben direkt zwei Versionen der Platine aufgebaut, nämlich für SPI und für I²C (**Bild 8** und **Bild 9**). Das Platinenlayout ist für beide Versionen geeignet, entscheidend ist dabei nur, welche Bauteile bestückt sind.

Und nun Ruff! - damit ins Ofenloch der SMD-Backstube! Im Nu waren die Platinen fertig, wurden noch genauestens (!) in Augenschein genommen, um eventuell missratene Exemplare auszusondern. Dann konnten die Jungs und Mädels aus dem Magazin den neuen Vorrat in die Regale stapeln, damit viele Sensorplatinen schnell an viele Kunden verschickt werden können. ⏮

(160297)





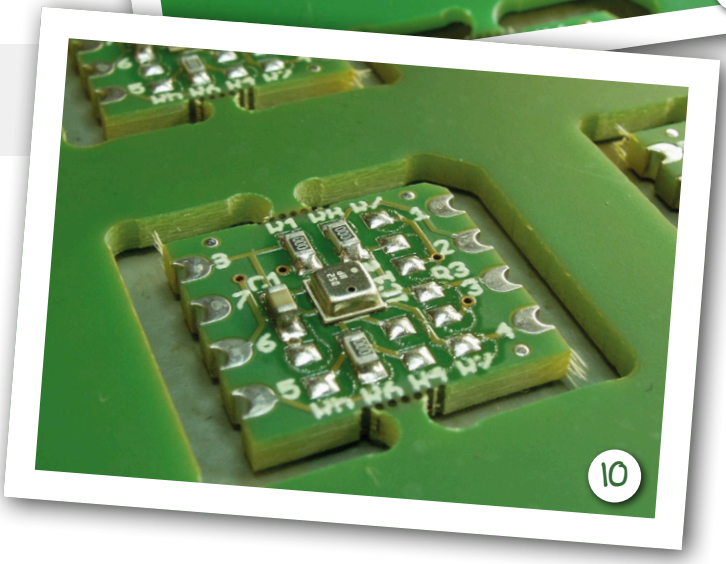
Weblinks

[1] www.elektormagazine.de/160109

[2] www.youtube.com/watch?v=HBWtqZro_fg

[3] www.eurocircuits.com

Im Internet findet sich ein kleines Video zur Anwendung des eC-stencil-fix: https://youtu.be/HBWtqZro_fg.



IM ELEKTOR-SHOP

→ 160109-2
Platine BME280-Shield

→ 160109-91
BME280-Sensorplatine (I²C-Version)

→ 160109-92
BME280-Sensorplatine (SPI-Version)

Von **Miroslav Cina** und
Dr. Thomas Scherer (Co-Autor Text)

Vor kurzem ist Miroslav Cina über einen Haufen alter Akkus von und für Notebooks gestolpert. Sie waren aussortiert, weil sie zur Versorgung der transportablen Rechner nicht mehr taugten. Neugierig hat er sie dann demontiert. Die Elektronik war unwichtig, und es stellte sich heraus, dass eine ganze Menge einzelner Zellen noch brauchbar waren. Um sie in eigenen Projekten einsetzen zu können, braucht es aber ein geeignetes Ladegerät. Und prompt hat er eines entwickelt.

Eigenschaften

- Lädt beliebige Li-Ion-Akkus mit einer Nennspannung von 3,6 V.
- Formatierungsmodus für Rettungsversuche tiefentladener Zellen.
- Entlademodus zur Ermittlung der realen Akkukapazität.
- Alle wichtigen Informationen verfügbar dank LC-Display.

Nicht nur Selbstbauprojekte, sondern auch käufliche Elektronik werden oft von Batterien oder durch NiMH-Akkus versorgt. Erstere muss man immer wieder neu kaufen, und letztere Energiequellen wiegen relativ viel. Außerdem sind beide gerade für Geräte mit höherem Strombedarf und längeren Lagerzeiten nicht optimal.
Deswegen stecken heutzutage in vielen Geräten Akkus auf Lithium-Basis, doch diese Technik kostet mehr und braucht mehr Aufwand. Wo es für AA-/AAA-Zellen und RC-Akkus in NiCd- oder NiMH-Technologie quasi in jedem Technikmarkt geeignete automatische Ladegeräte für wenig Geld gibt, sieht das für Lithium-Akkus eher mau aus. Der Grund dafür ist, dass Li-Ion-Akkus für gewöhnlich (wie bei Laptops oder Akkuschraubern) von Hause aus in Geräte eingebaut sind, und daher eine individuell geeignete Ladeelektronik eingebaut ist beziehungsweise das passende Ladegerät beiliegt.

Der Bedarf nach Universal-Ladegeräten ist auch daher gering, weil es keinen Markt für einzelne Lithium-Zellen gibt, die einfach in Geräte eingelegt und an der Kasse von Aldi, Lidl & Co. zum Mit-

Li-Ion-Akkulader

Ladegerät mit korrekten Kennlinien

nehmen bereitliegen. Lithium-Zellen werden in weniger stark reglementierten Formen und vor allem zum festen Einbau in Geräte, also mit Lötfahnen oder Schraubanschlüssen hergestellt und gehören aufgrund ihrer Empfindlichkeit, des erforderlichen anspruchsvollen Umgangs damit und gegebenenfalls sogar wegen ihrer Gefährlichkeit nicht in die Hände von Laien. Insbesondere die verbreiteten LiPo-Typen mit besonders hohem Leistungsgewicht sind auch mechanisch leicht verletzbar und brennen dann gerne. Hinzu kommt natürlich die bekannte Tatsache, dass eine NiMH-Zelle eine Nennspannung von 1,2 V aufweist. Bei Lithium-Technik ist mit einer Nennspannung von 3,6 V zu rechnen. Die stabileren LiFePo4-Typen mit ihrer Nennspannung von 3,3 V kommen in Bohrmaschinen, Handstaubsaubern, Smartphones und Notebooks praktisch nicht vor.

Li-Ion-Akkus

Unabhängig davon ob man Akkuzellen aus alten Akkublöcken wie in **Bild 1** ausbauen und weiter nutzen oder Li-Ion-Zellen neu kaufen möchte: Man braucht das passende Ladegerät. Dabei gibt es eine ganze Reihe Aspekte zu beachten, denn ganz so einfach wie bei NiMH-Akkus ist das Laden nicht. Es gibt Unterschiede in vielen Aspekten.

Zunächst eine Warnung: Lithium-Zellen darf man anders als Blei-, NiCd- oder NiMH-Zellen nicht ohne zusätzliche Maßnahmen in Serie schalten. Warum das so ist, steht im **Kasten Balancing**. Das hier beschriebene Ladegerät vermeidet diese Komplikationen, denn es ist für das Laden einer einzigen Lithium-Zelle mit einer Nennspannung von 3,6 V optimiert. Diese Nennspannung ist für praktisch jede Lithium-Chemie passend – mit Ausnahme der LiFePo4-Technologie.

Neben der Nennspannung von 3,6 V

INFOS ZUM PROJEKT

Lithium | Akkus | Ladegerät | Akku-Recycling

Einsteiger
➡ Fortgeschrittener
Experte

Etwa 3 Stunden

FTDI-USB/Seriell-Kabel
PICkit-Programmier-Schnittstelle (beide optional)

Rund 40 Euro

Bild 1. Konvolut ausgemusterter Laptop-Akkus mit ausgebauten Zellen. Viele davon lassen sich noch lange für andere Zwecke einsetzen.

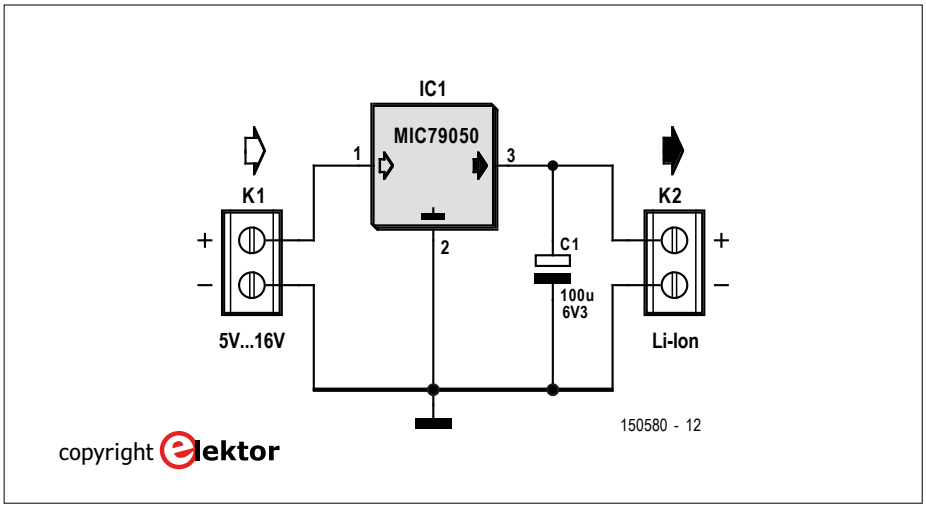


Bild 2. Die Minimalschaltung eines Ladegeräts für Lithium-Akkus kommt mit lediglich einem einzigen dreibeinigen IC (plus Elko) aus.

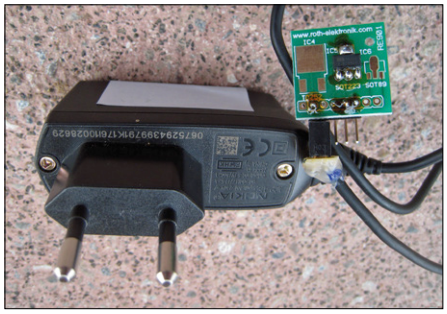


Bild 3. Zusammen mit einem Steckernetzteil hat sich Miroslav Cina (mit Hilfe der Schaltung von Bild 2) ein erstes funktionierendes Li-Ion-Ladegerät gebaut. Nachteil: Man sieht nicht, was gerade los ist.

gibt es noch einige andere „typische“ Werte, die man beim Bau eines Ladegeräts berücksichtigen muss. Für die Zellen gilt je nach Typ eine minimale Spannung von 2,75 bis 3 V und eine Maximalspannung von 4,2 V. Liegt die Zellenspannung beim Lagern, Entladen und Laden nicht in diesem Bereich, dann muss man mit Schädigungen rechnen. Nun ist man beim „Recyceln“ einzelner Zellen ja nicht unbedingt auf die volle Kapazität angewiesen. Für viele Zwecke tun es auch leicht angeschlagene Zellen mit reduzierter Kapazität. Man kann also auch versuchen, zu tief entladene Zellen wieder einsatzfähig zu machen. Dazu dürfen solche Exemplare aber nicht sofort mit dem vollen Ladestrom traktiert werden. Bei Spannungen unter 2,7 V müssen die Zellen mit einem Minimal-Ladestrom von einigen wenigen mA zunächst einmal „formatiert“ werden. Ab 2,7 V kann es dann richtig losgehen. Dieser Prozess kann aber unter Umständen einige Stunden dauern. Sollte eine Zelle nach vielen Stunden immer noch nicht diese Schwelle überschreiten, dann empfiehlt sich ein endgültiges Recyceln im Sinne der Entsorgung. Gleiches gilt für Zellen, die stabil 0,0 V aufweisen oder gar leicht negativ gepolt sind (kommt bei Serienschaltungen vor). Hier braucht man das Formatieren gar nicht erst zu versuchen.

Laderegeln

Ein wichtiger Wert für Lithium-Akkus ist der Wert „**C**“. Hier handelt es sich nicht etwa um die Abkürzung des englischen Begriffs „capacity“, es ist ein schlichter terminus technicus, der als Faktor den Strom relativ zur Kapazität angibt. Die Formel lautet: C = Kapazität in Ah / 1 h.

Ein maximaler Ladestrom von 1 C bedeutet bei einer Zelle mit einer Kapazität von 2.300 mAh dann einen maximalen Strom von 2,3 A. Eine Ladung mit einem konstanten Strom von 1 C ist übrigens bei vielen Zellen machbar. Spezielle oder ältere Zellen verkraften gelegentlich auch nur 0,5 C oder etwas weniger. Schnellladefähige Typen kommen sogar auf bis zu 5 C – sind also schon in 20 Minuten voll geladen. Weniger Ladestrom als der Maximalwert ist natürlich immer möglich.

Auch beim Entladen spielt dieser Faktor eine Rolle, aber man hat es hier auch mit höheren Werten zu tun. 2 C sind quasi normal und die in den Autos von Tesla eingesetzte Panasonic-Zelle vom Typ 18650B kommt auf diesen Wert, falls Sie mal einen Tesla ausschlachten wollen ;-)

Es geht aber auch mehr: LiPos kommen häufig auf 5 C oder 10 C und Spezialzellen können kurzfristig auch über 20 C liefern. Für das Laden spielt das aber eine untergeordnete Rolle.

Aus diesen Kennwerten ergeben sich direkt die Anforderungen an ein Lithium-Ladegerät: Es muss bei Spannungen unter 2,7 V mit nur wenigen mA laden, ab dieser Schwelle dann mit einem bestimmten höheren Konstantstrom weiterladen, bis die Schlussspannung von 4,2 V erreicht ist. Ab da wird einfach die Spannung von 4,2 V konstant gehalten und der Ladestrom reduziert sich automatisch mit steigendem „Füllstand“ der Zelle. Unterschreitet der sich nun ergebende Ladestrom irgendwann einen Minimalwert von z.B. 0,05 C, wird die Ladung beendet und der Akku als voll betrachtet. Wichtiges Extra: Gerade geschädigte Akkus sind möglicherweise so ausgelaugt, dass der zum Abschalten führende Strom nie erreicht wird. Hier muss dann eine Zeitabschaltung ein ewiges Laden verhindern.

Integrierte Lösung

Da Lithium-Akkus ja vorwiegend in Geräte eingebaut sind, ist es kein Wunder, dass die Halbleiter-Industrie bereits allerhand integrierte Laderegler im Programm hat, um den Geräte-Produzenten das Leben zu erleichtern. Eine der einfachsten integrierten Ladelösungen überhaupt ist das dreibeinige IC MIC79050S von Microchip. Ein damit aufgebautes Ladegerät ist tatsächlich so einfach wie in **Bild 2** dargestellt. Dieser Laderegler

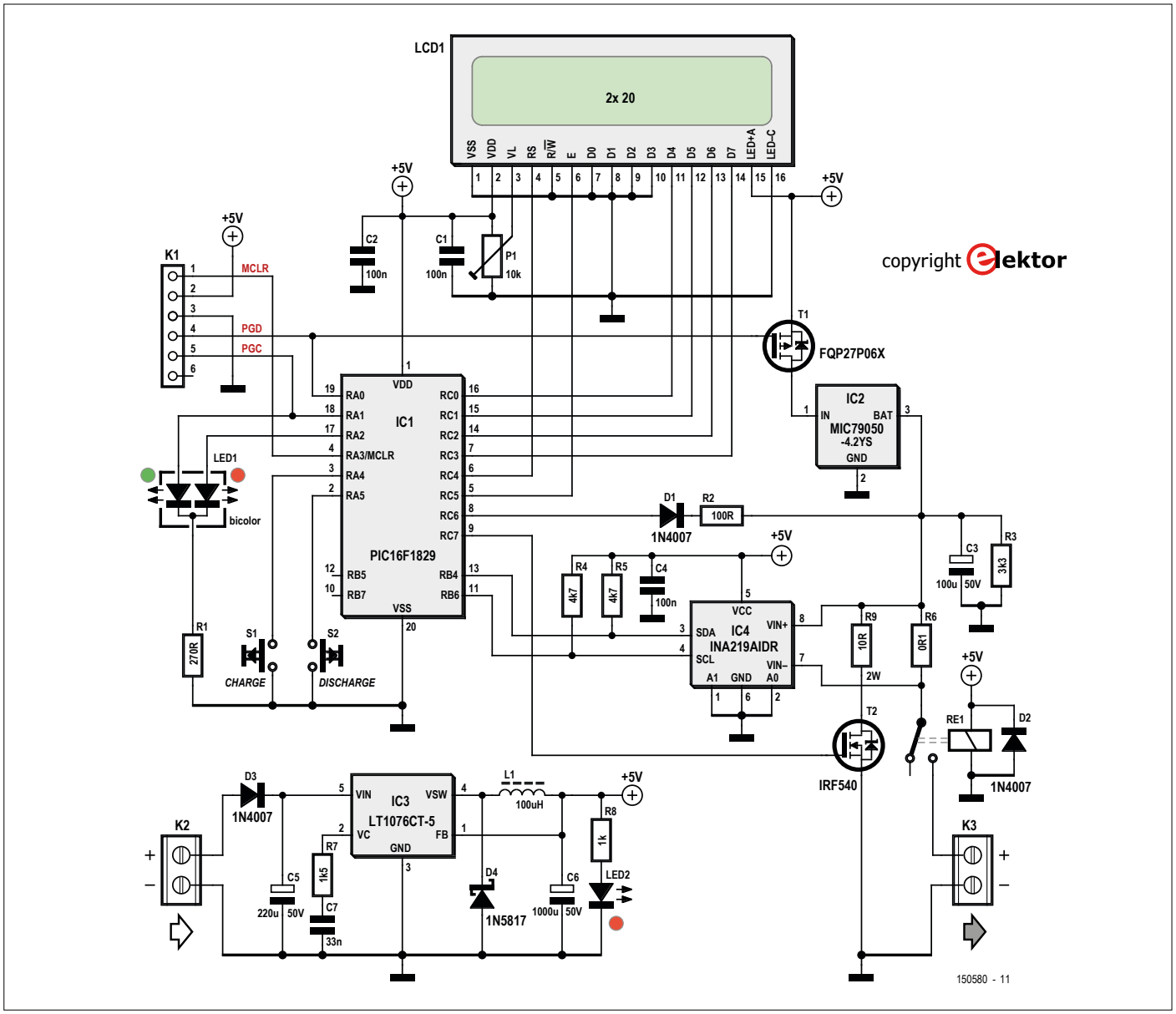


Bild 4. Mit Hilfe eines Mikrocontrollers und etwas mehr Elektronik wird aus dem Simpel-Laderegler von Bild 2 ein richtig luxuriöser Lithium-Lader.

stellt an seinem Ausgang eine stabilisierte Spannung von 4,2 V mit einer (notwendigen) Toleranz von 0,75 % zur Verfügung. Sein Ladestrom beträgt maximal 0,5 A – nicht viel, aber daher auch für die allermeisten Zellen geeignet. Eine übliche Zelle mit 2...3 Ah ist also in akzeptablen 4...6 Stunden aufgeladen.

Außer einer eingebauten Übertemperatursicherung kann das IC nicht viel. Dafür ist es im lötfreundlichen Gehäuse SOT223-3 verfügbar und braucht am Eingang nicht mehr als eine Spannung im Bereich 5...16 V, die mit 0,5 A belastbar ist. Empfehlenswert ist dabei, im unteren Spannungsbereich zu bleiben, denn der Linearregler verheizt sonst viel Energie und wird entsprechend heiß. Miroslav Cina baute sich daher zum Test auf Tauglichkeit einen simplen Lithium-Lader auf, bestehend aus einem alten Steckernetzteil eines Nokia-Handys (5,7 V / 0,8 A) und dem IC auf einer Experimentierplatine (siehe **Bild 3**). Der Test ergab grünes Licht.

Das Einfachst-Ladegerät funktioniert zwar, hat aber gerade für Elektroniker einen entscheidenden Nachteil: Irgendwann ist der Akku wohl voll, aber man weiß nicht wann und sieht nicht, was dabei passiert. Hierzu ist dann schon etwas mehr Elektronik nötig, weshalb sich Miroslav an die Entwicklung eines „richtigen“ Ladegeräts machte.

Luxusversion

Mehr Luxus heißt heute, dass eine Grundfunktion via Mikrocontroller aufgepeppt wird. So auch hier: **Bild 4** zeigt die komplette Schaltung eines komfortablen Ladegeräts für handelsübliche Li-Ion-Zellen. Der Mikrocontroller vom Typ PIC16F1829 steuert nicht nur das Laden an sich, sondern misst die Spannung und den fließenden Strom, steuert eine zweifarbige Status-LED, fragt zwei Taster ab und zeigt alle wichtigen Daten auf einem LCD an.

Das ist aber noch nicht alles: Damit nicht allzu viel Energie verheizt wird, ist mit IC3 auch noch ein Schaltregler eingeschleift. Doch keine Angst, die zugehörige Spule muss man nicht selbst wickeln, denn L1 gibt es als günstige Festinduktivität. D3 dient als Verpolschutz. Da IC3 für 5 V Ausgangsspannung eine Eingangsspannung zwischen 8 V und 60 V

benötigt, genügt zur Versorgung der Schaltung ein simples, unstabilisiertes Steckernetzteil mit etwa 5 W Belastbarkeit. Seine Ausgangsspannung sollte zwischen 9 V und 36 V liegen. Ideal wären 12 V und 0,5 A. Bei D4 sollte es unbedingt eine schnelle Schottky-Diode mit 1 A Belastbarkeit sein. Ein simpler Ersatz durch eine Silizium-Diode aus der Grabbelkiste ist keine gute Idee. LED2 leuchtet, wenn die 5 V von IC3 anliegen.

IC1 erfasst mit den I/O-Pins RA4 und RA5 den Status der beiden Taster S1 und S2. Das klappt so spartanisch, weil der Controller an seinen Eingängen interne Pull-ups aktiviert hat. RA1 und RA2 steuern die zweifarbige LED (siehe **Kasten Bi-Color-LED und Status**). LCD1 wird im klassischen Halb-Byte-Verfahren angesteuert und bietet je nach Typ zwei Zeilen mit je 20 oder 24 Zeichen. P1 ist für die Kontrasteinstellung gedacht. Die Hintergrundbeleuchtung ist direkt an +5 V angeschlossen (Pins 15 und 16). Man muss also darauf achten, ein Display mit integriertem Vorwiderstand zu nehmen, falls man ein anderes als das in der Stückliste vorgeschlagene Modell wählt. T1 ist ein P-Kanal-Power-MOSFET und schaltet die Versorgung des eigentlichen Ladereglers IC2 ein und aus.

Die Strommessung wird mit Hilfe des darauf spezialisierten ICs INA219 über einen niederohmigen Shunt (R6) vorgenommen. IC4 ist per I²C mit IC1 verbunden. Für IC4 genügt hier übrigens die preiswertere und nicht ganz so genaue Ausführung INA219B.

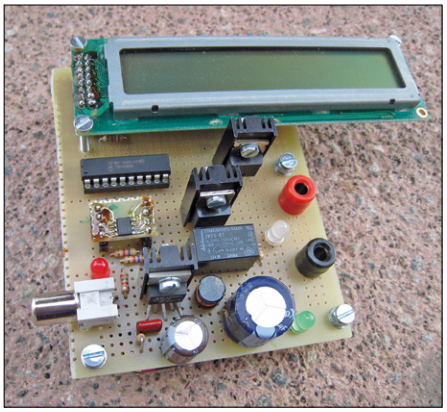


Bild 5. So sah der erste Lochraster-Aufbau des Luxus-Laders von Miroslav aus.

Über T2 kann der Lastwiderstand R9 an den Akku gelegt werden, um ihn zu entladen. Dank einer mit S2 startbaren Entladefunktion kann vollautomatisch die reale Kapazität eines (voll geladenen) Prüflings ermittelt werden.

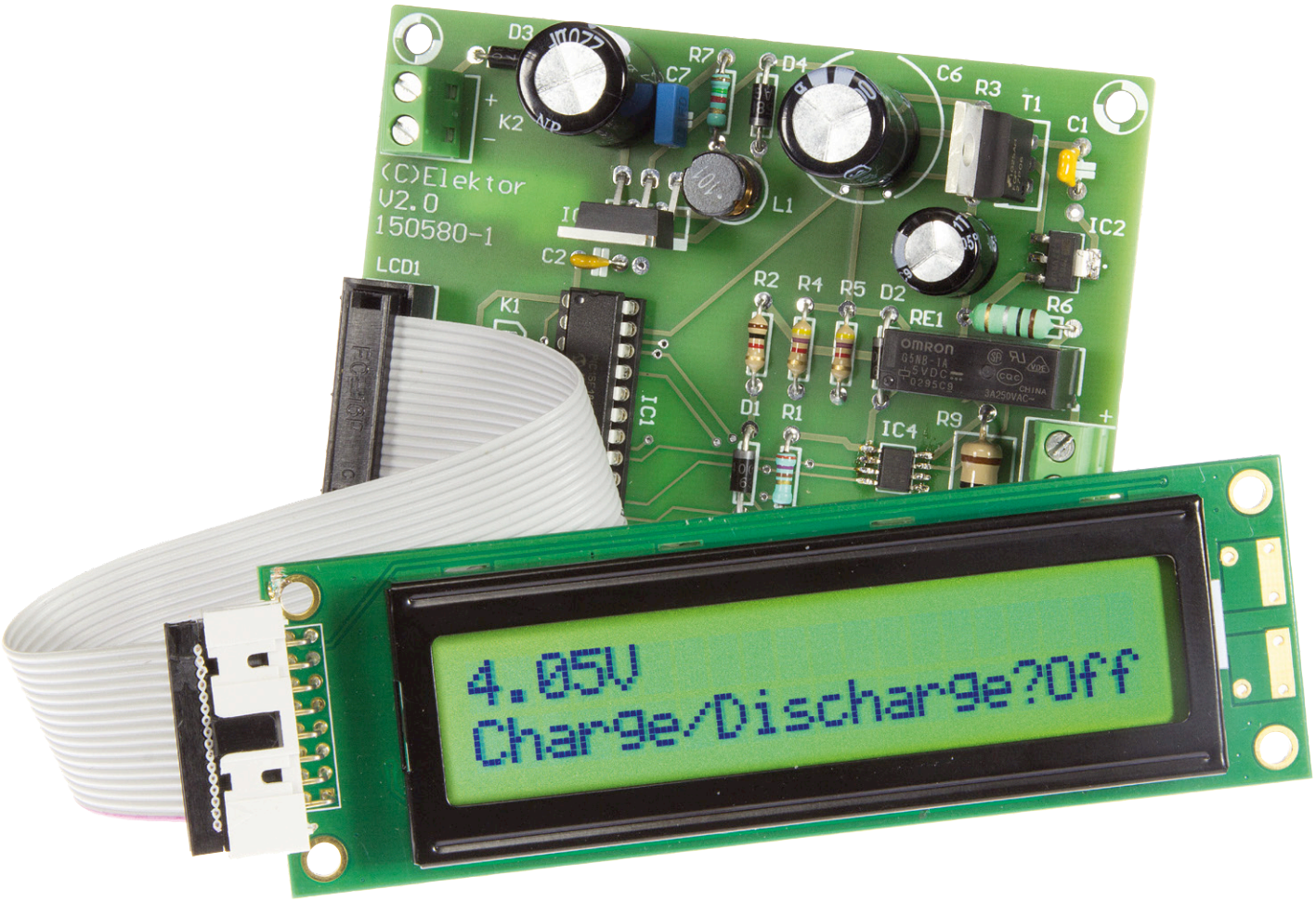
Die „Formatierung" eines zu tief entladenen Akkus erfolgt über einen High-Pegel an Pin RC6 von IC1. Der Strom wird über R2 und den Innenwiderstand des MCU-Pins begrenzt und D1 verhindert einen Rückstrom. Die Höhe des Formatierungsstroms hängt von der Zellenspannung ab. Ein Beispiel: Bei 2,3 V liegen 5 V abzüglich der Durchlassspannung von D1 und der Zellenspannung = 2 V an R2. Es fließt folglich ein maximaler Strom von 20mA. Mehr als 25 mA kann die MCU nicht liefern.
Bleibt noch zu klären, wozu RE1 eigentlich da ist, denn es wird ja nicht vom Mikrocontroller angesteuert. Des Rätsels Lösung ist ganz einfach: Das Relais ist nur dann angezogen, wenn das Ladegerät versorgt wird. Zieht man den Stecker, wird ein eventuell angeschlossener Akku abgetrennt und so nicht versehentlich „rückwärts" über IC3 und IC4 entladen, was unabsichtliche Tiefentladungen zuverlässig verhindert.

Software

Kein Mikrocontroller ohne Firmware. Hier hat sie folgende Aufgaben zu erledigen: Aktuellen Zustand ermitteln, Strom und

Balancing

Anders als bei der Chemie üblicher Akku-Technologien reagiert der Inhalt von Lithium-Zellen empfindlich auf Überladung und Tiefentladung, was den Akku irreversibel schädigt. In einer Serienschaltung wird ohne explizite elektronische Überwachung (eben dem Balancing) die Zelle mit der kleinsten Kapazität beim Laden irgendwann „übervoll" sein (die Zellenspannung steigt über den zulässigen Wert), während die anderen Zellen noch weiter geladen werden wollen. Neben der Temperaturerhöhung und der Feuergefährlichkeit mancher Typen hat die daraus folgende Schädigung der betreffenden Akkuzelle dann eine weitere Reduktion ihrer Kapazität zur Folge. Genauso beim Entladen: Die Zelle mit der kleinsten Kapazität rutscht dabei ohne Überwachung leicht in kritische Unterspannungsbereiche, was ebenfalls einen Kapazitätsverlust bedeutet. Nach mehreren Lade/Entladezyklen schaukelt sich das Problem also schnell auf.
Deshalb gilt die eiserne Regel: Lithium-Zellen niemals ohne sogenannten Balancer oder explizite Akku-Management-Elektronik in Serie schalten. Das gilt für den normalen Betrieb (= Entladen) genauso wie für das Laden. Geeignete Balancer gibt es für unterschiedliche Zellenzahlen, Kapazitäten und unterschiedliche Lithium-Chemie fertig zu kaufen. Das Thema Balancing ist komplex und würde den Rahmen dieses Artikels sprengen. Auch in Elektor gab es schon Artikel dazu, die man per Suchfunktion auf www.elektormagazine.de findet. Für viele Zwecke genügt aber schon die Verwendung einer einzigen Zelle, und dann kann man Balancer-Fragen getrost ignorieren.



Lithium-Akkus recyceln - dank universellem Li-Ion-Ladegerät

Spannung messen, Taster abfragen, LED1, T1 und T2 ansteuern, die Zeit erfassen und relevante Informationen auf dem LC-Display ausgeben. Die Firmware wurde in Assembler geschrieben und ist wie immer sowohl als Source-Code als auch als Hex-Datei im Download [1] kostenlos erhältlich. Nachfolgend eine kurze Erläuterung relevanter Teile:

Die Software initialisiert zunächst den Zeit-Zähler (mit Interrupts und TIMER2); dann wird die Hauptschleife gestartet. Für den Ladeprozess ist die Routine *charge_main* zuständig. In der Hauptschleife werden die aktuellen Werte von Strom und Spannung via INA219 per I²C ausgelesen und ausgewertet. Falls am Anfang eine Zellenspannung unter 2,7 V ansteht, wird zunächst die Formatierung vorgenommen. T1 bleibt derweil

Bi-Color-LED und Status

Mit Hilfe der zweifarbigen LED1 werden die Betriebszustände des Li-Ion-Akkuladers signalisiert. Dabei ergeben sich folgende Zuordnungen:

- LED1 = **aus**. Das Ladegerät ist nicht aktiv. Der Akku wird weder geladen noch entladen.
- LED1 = **orange** (R- und G- leuchten beide). Formatierungsprozess läuft.
- LED1 = **rot**. Ladephase. Der Ladestrom ist >200 mA.
- LED1 = **grün**. Ladephase. Der Ladestrom ist 80...200 mA.
- LED1 = **rot blinkend**. Entladephase.

Daraus folgt, dass sich bei eingeschaltetem Ladegerät und angeschlossenem Akku erst einmal gar nichts tut und lediglich die LED2 leuchtet. Nach Betätigung von S1 leuchtet die LED1 (solange die Zellenspannung unter 2,7 V liegt) zunächst orange. Oberhalb dieser Spannung leuchtet sie rot, wenn der Strom >0,2 A beträgt, und sie wird grün, wenn er darunter fällt. Unterhalb eines Stroms von 80 mA erlöscht die LED und die Ladung ist beendet.

Bei Betätigung von S2 blinkt die LED solange, bis die Entladeschlussspannung von 3 V erreicht ist. Alle Prozesse kann man jederzeit durch gleichzeitige Betätigung beider Taster abbrechen.

gesperrt. Steigt die Spannung während der Formatierung über 2,69 V, wird die Formatierung beendet und via T1 und MIC79050 der normale Ladevorgang gestartet.

Beim normalen Laden werden Spannung und Strom periodisch gemessen. Die Hauptschleife wird beendet, wenn der Ladestrom unter 80 mA sinkt oder wenn die Gesamtladezeit von 30 Stunden überschritten wird. Beide Werte können in der Firmware geändert werden.

Der Wert für die maximale Ladedauer findet sich in der Subroutine *mm02* an der Stelle:

```
movlw D'030' ;maximal charging
 time (hours)
```

Der Wert für den Minimalstrom steckt im Unterprogram *curr_an* an der Stelle:

```
xorlw D'080' ;keep charging above
 080 mA
```

Man kann auch die Schwelle für die Umschaltung von hohem zu niedrigem Ladestrom in der gleichen Subroutine ändern.

```
xorlw D'200' ;green LED off above
 200 mA
```

Die Subroutine *discharge_main* ist für die Entladung zuständig. Die Entladung stoppt, wenn die Zellenspannung unter 3 V sinkt. Bei der Entladung wird nicht etwa die Zeit bewertet, sondern als rele-

STÜCKLISTE

Widerstände:
(alle 5 %, ¼ W, 250 V)
R1 = 270 Ω
R2 = 100 Ω
R3 = 3k3
R4,R5 = 4k7
R6 = 100 m, 1 W
R7 = 1k5
R8 = 1 k
R9 = 10 Ω, 2 W

Kondensatoren:
C1,C2,C4 = 100 n / 50 V, keramisch, RM 2/10"
C3 = 100 µ / 50 V, Elko, stehend, RM 3,5 mm, ⌀ 8 mm
C5 = 220 µ / 50 V, Elko, stehend, RM 5 mm, ⌀ 10 mm
C6 = 1000 µ / 50 V, Elko, stehend, RM 7,5 mm, ⌀ 16 mm
C7 = 33 n / 50 V, keramisch, RM 2/10"

Spule:
L1 = 100 µ, 190 mΩ, 900 mA, stehend, MCSCH895-101KU

Halbleiter:
D1..D3 = 1N4007
D4 = 1N5817 *
LED1 = bi-color, 5 mm, gemeinsame Kathode
LED2 = rot, 3 mm
T1 = FQP27P06X, P-Kanal, 70 mΩ
T2 = IRF540NPBF, N-Kanal, 44 mΩ
IC1 = PIC16F1829-I/P, programmiert*
IC2 = MIC79050-4.2YS, TO223-3
IC3 = LT1076CT-5
IC4 = INA219AIDR*

Außerdem:
LCD1 = LCD, mit Hintergrundbeleuchtung, z.B. MC22005A6W-GPTLY von Midas *
Stiftleiste, 1 x 6, stehend, RM 1/10"
2 zweipolige Schraubklemmen für Platinenmontage, RM 2/10"
2 Taster für Platinenmontage, Schließer, 6 x 6 mm
Relais, 5 V, 1-pol. Schließer, 5 A, z.B. Omron G5NB1A5DC
20-pol. IC-Fassung, DIP, für IC1
Platine 150580-1 V2.0 *

* siehe Text

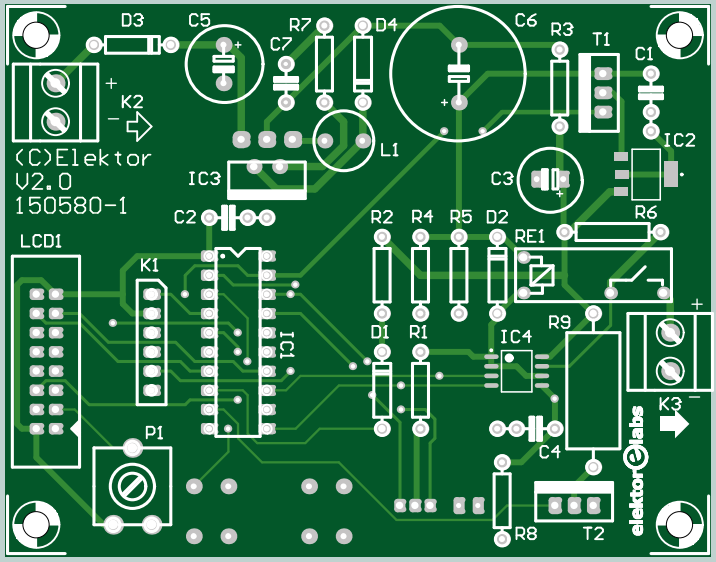
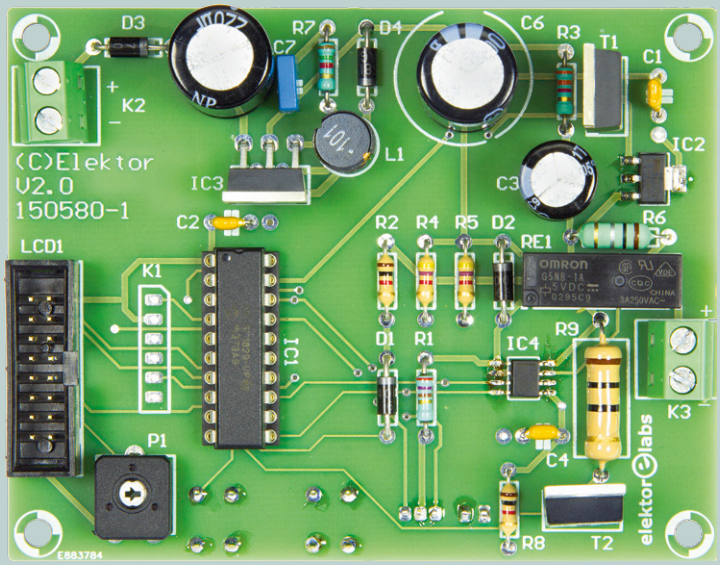


Bild 6. Dank Platine ist der Aufbau des luxuriösen Lithium-Laders ein Kinderspiel.



vante Größe die vom Akku gelieferte Strommenge = Kapazität in mAh.

Es gibt kein Timeout. Die eigentliche Messung erfolgt in der Subroutine *inc_q* in mAs.

Bedienung

Nach dem **Einschalten** wird zuerst kurz die Firmware-Version angezeigt; danach kommen Informationen über den angeschlossenen Akku. Die Darstellung ist hier für das Display mit 2 x 20 Zeichen wiedergegeben:

```
U = 3.10V
CHARGE? / DISCHARGE?
```

Jetzt könnte man mit S1 die Ladung oder mit S2 die Entladung starten.

Beim **normalen Laden** (Zellenspannung >2,7 V) zeigt das LCD zunächst zum Beispiel solche Infos:

```
4.16V/+203MA/1736MAH
05:02:29          25
```

In der ersten Zeile stehen aktuelle Spannung und aktueller Strom sowie die gesamte bisher eingeladene Strommenge in mAh. In der zweiten Zeile finden sich links die bisher verstrichene Zeit in hh:mm:ss und rechts die noch bleibenden Stunden bis zum Timeout (hier 25 h).

Wenn nun der Strom unter die **Schwelle von 200 mA** sinkt, dann zeigt das Display:

```
4.16V/+199MA/1758MAH
05:05:29-00:05:24   25
```

In der zweiten Zeile kommt in der Mitte die verstrichene Zeit des Ladens mit reduziertem Strom hinzu.

Ist die Zelle **komplett geladen**, erlischt LED1 und es ist zum Beispiel die folgende Anzeige zu sehen:

```
4.16V      1807MAH
05:05:29-00:10:32OFF
```

In der zweiten Zeile sieht man jetzt am Ende statt des Timeouts die Information „Off“.

Sollte die Zellenspannung zu gering sein, startet der **Formatierungsprozess** mit folgender Anzeige:

```
1.28V/+016MA/0000MAH
FORMAT   00:01:43 30
```

Neben Zellenspannung und fließendem Strom in der oberen Zeile werden sich bei den geringen Strömen allenfalls minimale Strommengen ergeben. In der unteren Zeile wird angezeigt, wie lange der Prozess schon dauert und wie lange die Ladung maximal noch insgesamt dauern könnte.

Beim **Entladen** via S2 wird der Akku über R9 entladen. Ein gut geladener Akku wird also zunächst einen Strom von rund 360 mA fließen lassen. Die Anzeige wird also etwa so aussehen:

```
3.72V/-364MA/0759MAH
00:46:10
```

In der oberen Zeile ist also hinter den aktuellen Werten von Spannung und Strom (negatives Vorzeichen = Entladung) die bisher entnommene Strommenge in mAh zu sehen. Die untere Zeile zeigt die verstrichene Zeit.

Aufbau und Schlussbemerkungen

Der Prototyp von Miroslav Cina auf einer Lochrasterplatine ist in **Bild 5** zu sehen. Mit Hilfe der Platine (**Bild 6**) aus dem Elektor Shop [1] ist der Aufbau aber bedeutend einfacher. Schließlich sind alle Bauteile bis auf das trotz SMD-Gehäuse leicht zu lötende IC2 vom bedrahteten Typ. IC1 ist fertig programmiert (mit der Firmware für ein 2x20-Zeichen-Display) erhältlich. Wer ein Display mit 2x24 Zeichen nutzen will, kann die alternative Firmware im Download verwenden. Wer ein alternatives Display einsetzen will, sollte darauf achten, dass es über einen HD44780-kompatiblen Controller verfügt. Wenn man alles richtig gemacht hat, dann müsste das Ergebnis etwa so aussehen wie der Prototyp aus dem Elektor-Labor (**Bild 7**).

Bei Anschluss eines Akkus muss man sorgfältig sein. Ein verpolter, geladener Akku kann nämlich durchaus IC4 beschädigen, da der Chip an seinen Messeingängen nur eine niedrige negative Spannung verkraftet. Wer das sicher verhindern will, der kann die Schutzschaltung von **Bild 8** zwischen Ladegerät und Akkuzelle schalten. Wird hier der Akku verkehrt angeschlossen, dann leitet die Diode und die Sicherung löst aus. Damit das schnell geht, sollte die Sicherung ein flinkes Exemplar sein. Und damit die Spannung auch im Fehlerfall möglichst niedrig bleibt, empfiehlt sich für D eine Schottky-Diode mit einer Belastbarkeit von mindestens 3 A.

Miroslav Cina hat seit dem Bau seines Li-Ion-Laders schon bei etlichen Geräten die NiMH-Akkus durch Lithium-Zellen ersetzt und deren Eigenschaften dadurch deutlich verbessert. Sein Ladegerät kommt also relativ oft zum Einsatz, und er kann daher sagen, dass es sich wirklich in der Praxis bewährt hat. Für Fragen, Anregungen und Verbesserungsvorschläge ist er jederzeit über die E-Mail-Adresse miroslav.cina@t-online.de erreichbar. ◀

(150580)

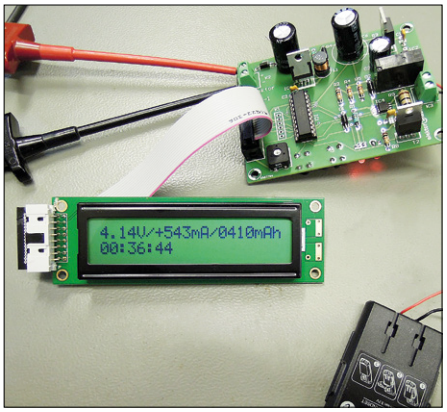


Bild 7. Der mit der Platine in Bild 6 realisierte Prototyp aus dem Elektor-Labor.

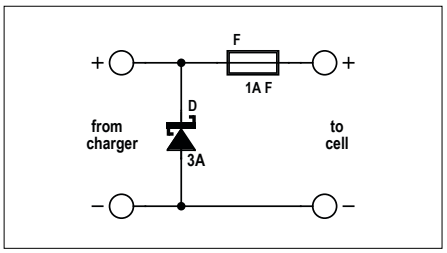


Bild 8. Diese simple Schutzschaltung verhindert Schäden durch versehentlich verpolte Akkus.

Weblink

[1] www.elektor.de/150580

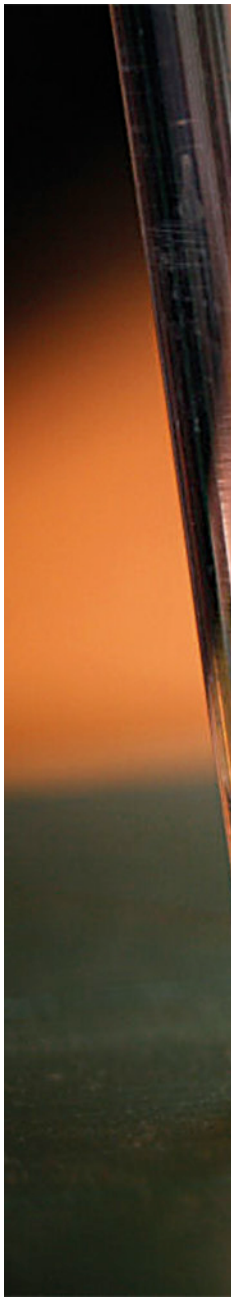
EINKAUFSZETTEL

→ 150580-1
Platine

→ 150580-41
Programmierter Controller, Version für 2x 20 Zeichen

ELEKTOR ETHICS

Brauchen wir ein Robotergesetz?



Die europäischen Gesetzgeber wollen die ethischen und rechtlichen Aspekte von Robotern und künstlicher Intelligenz EU-weit regeln. Sollte dieses „Neuland“ bereits jetzt reguliert werden? Oder sollte man die neue Technologie erst reifen lassen, bevor sie von Gesetzen beschränkt wird?

Von **Tessel Renzenbrink** (Niederlande)

Im vergangenen Februar hat das Europäische Parlament (EP) die Europäische Kommission aufgefordert, Regeln für Robotik und KI vorzulegen. Die Entschließung besteht aus etwa 28 Seiten Empfehlungen, die einen Rechtsrahmen abstecken sollen. Dabei geht es um eine breite Palette von Gesichtspunkten wie Innovation, Normung, ethische Grundsätze, Beschäftigung, Bildung und Haftungsregeln. Wir haben zusammengefasst, was das Parlament zu den folgenden Themen sagt.

Innovation

Europa ist ein Vorreiter in Sachen Künstlicher Intelligenz und der Forschung und Entwicklung von Robotik. Damit dies so bleibt, sollte die EU die Forschungsbudgets erhöhen und die offene wissenschaftliche und verantwortungsbewusste ethische Innovation fördern. Zweitens: Eine solide digitale Infrastruktur stellt die Basis der Entwicklung dieser Technologien dar. Die EU sollte sich daher für einen allgegenwärtigen und universellen Netz-Zugang und die Wahrung des Prinzips der Netzneutralität einsetzen. Und schließlich ist eine Interoperabilität von Systemen für Echtzeit-Datenflüsse unerlässlich, die es Robotern und KI ermöglichen, flexibler und letztendlich autonom zu werden. Daher sollten offene Standards und Transparenz gefördert werden, um eine Einbindung in proprietäre Systeme zu verhindern, die die Interoperabilität einschränken würden.

Standardisierung

Standardisierung und Interoperabilität sind entscheidend zur Förderung eines gesunden Wettbewerbs im Bereich Robotik und KI. Die EU sollte daher zusammen mit europäischen und internationalen Normungsgremien an der internationalen Harmonisierung der technischen Normen arbeiten. In der EU sollten EU-weite Regeln die Zersplitterung des Binnenmarktes und damit auch die Entwicklung inkompatibler Systeme in den Mitgliedstaaten verhindern. Maker und Erfinder werden sich freuen zu erfahren, dass das EP die Bedeutung des rechtmäßigen Reverse-Engineerings betont und „bastlerfreundliche“ Zonen für Experimente mit Robotern gestatten will.

Ethische Richtlinien

Asimovs Robotergesetze können nicht in Maschinencode umgewandelt werden. Solche Gesetze sollten an Entwickler, Ingenieure, Produzenten und Betreiber von Robotern in der EU gerichtet werden. Ein Verhaltenskodex für alle, die auf diesem Gebiet tätig sind, würde sie auf vier Grundprinzipien verpflichten:

1. Roboter sollten im besten Interesse des Menschen handeln.
2. Roboter sollten einem Menschen nicht schaden.
3. Menschen sollten frei sein, eine fundierte Entscheidung über die Bedingungen der Interaktion mit Robotern zu treffen.
4. Der Gewinn der Robotik insbesondere in der häuslichen Nutzung und im Gesundheitswesen sollte fair verteilt werden.

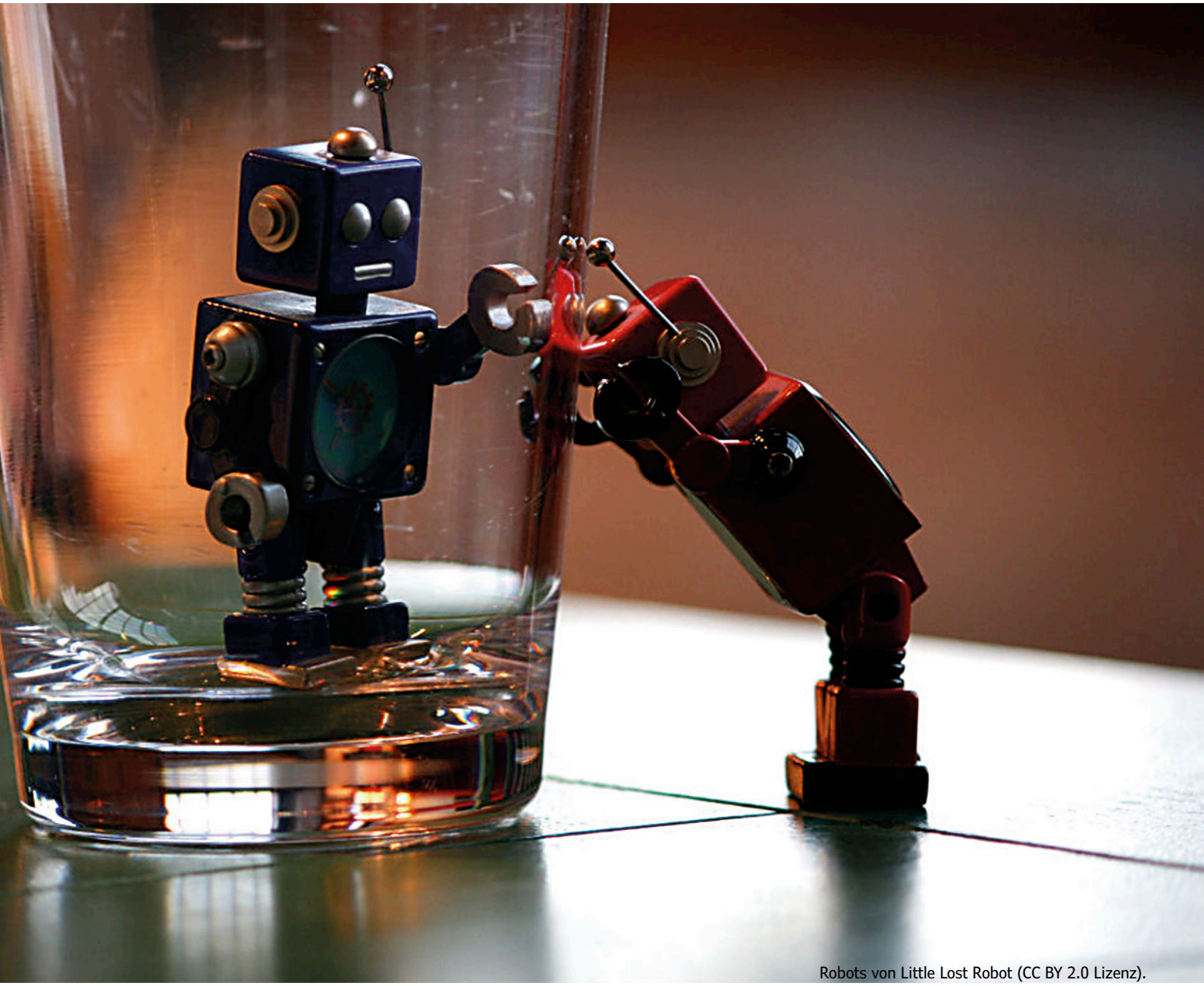
Daneben drängt das EP darauf, die potenziellen Risiken der KI und der Robotik für Menschen und die Gesellschaft als Ganzes zu beachten. Entwicklung und Nutzung von Robotern sollte daher stets den Werten entsprechen, die im Vertrag über die Europäische Union verankert sind: Menschenwürde, Gleichheit, Recht und Gerechtigkeit, Nichtdiskriminierung, Einverständnis und Datenschutz.

Arbeit und Bildung

Der am meisten in der Öffentlichkeit diskutierte Aspekt der Robotik ist der Verlust von Arbeitsplätzen aufgrund der Automatisierung. Wenn viele Arbeitsplätze verloren gehen, ohne dass Ersatz geschaffen werden kann, müssen wir die Organisation der Gesellschaft überdenken. Das Prinzip von Arbeit als Mittel zur Verteilung von Reichtum könnte nicht mehr praktikabel sein. Das EP fordert deshalb die Kommission auf, diesen Trend zu beobachten und mögliche Konsequenzen für die Systeme der sozialen Sicherheit in den Mitgliedstaaten zu analysieren. Im Hinblick auf das Bildungswesen warnt das EP vor einem wachsenden Mangel an IT-Fachkräften. 90 % aller Arbeitsplätze werden zumindest digitale Grundfähigkeiten erfordern. Das Parlament rät der EU, flexible Ausbildungs- und Bildungssysteme zu entwickeln, um Europa auf die zunehmende Digitalisierung und Automatisierung der Wirtschaft vorzubereiten.

Verantwortlichkeit

Ein weiteres oft diskutiertes Thema ist das der Haftung. Wer ist verantwortlich für den Schaden, den ein autonomes oder



Robots von Little Lost Robot (CC BY 2.0 Lizenz).

selbstlernendes System verursacht? Der Produzent? Der Besitzer? Der Roboter selbst? Solange dieses Problem ungelöst ist, könnte es Innovationen behindern, weil sowohl Produzent als auch Endverbraucher Rechtssicherheit benötigen. Deshalb fordert das Parlament die Kommission auf, ein Gesetzgebungsinstrument zur Frage der Haftung vorzuschlagen.
Schließlich fordert das EP eine Europäische Agentur für Robotik und KI als Kompetenzzentrum, um die oben genannten Ziele umzusetzen.

Öffentliche Debatte

Die Kommission ist nicht verpflichtet, dem Vorschlag des Parlaments zu folgen, müsste eine Ablehnung aber begründen. Zum Redaktionsschluss dieses Artikels hatte die Kommission noch nicht geantwortet. Sollte die Kommission die Entschließung akzeptieren oder zurückweisen?
Gegner sagen, es wäre viel zu früh für irgendeine gesetzliche Regelung. Das Fachgebiet ist immer noch jung und wir wissen zu wenig darüber, wie sich die Robotik entwickeln wird. Sie wehren sich auch dagegen, KI und Robotik, zwei verschiedene Felder, über einen Kamm zu scheren. Auch im Bereich der Robotik gibt es sehr große Unterschiede. Ein Staubsaugerroboter muss offensichtlich weniger beaufsichtigt werden als ein Pflegeroboter für ältere Menschen. Der stärkste Einwand ist jedoch, dass zu viel gesetzliche Regelung die Innovation erstickt.
Die Befürworter differenzieren. Sie argumentieren, dass ein gemeinsamer Rechtsrahmen die Innovation tatsächlich eher fördern dürfte, erstens, weil er Rechtssicherheit für Hersteller und Endverbraucher schafft und zweitens, weil er die technologischen Standards in den Mitgliedsstaaten harmonisiert.
Ob wir einen gesetzlichen Rahmen für die Robotik brauchen, ist also umstritten. Aber es ist höchste Zeit, darüber nachzudenken. Bevor uns die nächste technologische Welle überrollt, brauchen wir eine öffentliche Debatte darüber, was wir mit Robotik und KI erreichen wollen. Wie unsere Gesellschaft von diesen neuen Technologien geprägt werden soll, muss demokratisch entschieden werden! ◀

(160334)

Willkommen in Ihrem E-SHOP

ELEKTOR EMPFIEHLT

MiniPro TL866 Universal Programmer

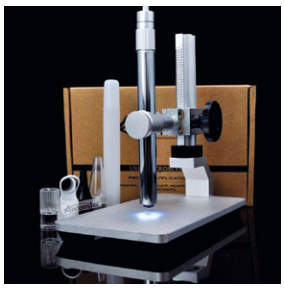
Heutzutage sind ja die meisten Mikrocontroller mit einem Flash-Programmspeicher ausgestattet, der „in-circuit" programmiert werden kann. Aber vor nicht allzu langer Zeit besaß fast jeder Hobby-Elektroniker ein Gerät, mit dem er (E)EPROMs, GALs oder Mikrocontroller programmieren konnte. Und dann am liebsten ein Modell, das für möglichst viele Typen geeignet war. Früher war ein solches Gerät für den Hausgebrauch beinahe unbezahlbar, doch heute steht mit dem MiniPro TL866A ein wahrer Alleskönner zu einem vernünftigen Preis zur Verfügung. Der TL866 unterstützt die Programmierung von über 14.000 verschiedenen Bauteilen, angefangen bei archaischen (E)EPROMs bis hin zu den neuesten Mikrocontrollern. Und das Gerät kann auch die Funktion von Logik-ICs (CMOS und TTL), DRAMs und SRAMs überprüfen.

Luc Lemmens (Elektor-Labor)

www.elektor.de/tl866a-programmer

Elektor-Bestseller

1. Andonstar USB-Mikroskop V160
   www.elektor.de/andonstar-v160



2. ESP8266-WLAN-Funkmodul
   www.elektor.de/150445-91
3. Raspberry Pi 3
   www.elektor.de/rpi-3
4. Swiss Pi
   www.elektor.de/swiss-pi
5. BBC micro:bit
   www.elektor.de/bbc-micro-bit
6. Das ESP8266-Praxisbuch
   www.elektor.de/esp8266-buch
7. Serielle Schnittstellen
   www.elektor.de/serielle-schnittstellen
8. Arduino-Sanduhr
   www.elektor.de/sanduhr

Serielle Schnittstellen

Die serielle Schnittstelle dient dem Datenaustausch zwischen Mikrocontrollern und Peripheriegeräten. Das Wissen um diese seriellen Schnittstellen erweitert die Einsatzmöglichkeiten von Mikrocontrollern und Peripheriegeräten ungemein. In diesem Buch werden die Grundlagen der Kommunikation für die drei seriellen Schnittstellen I²C, SPI und 1-Wire vorgestellt.

Mitgliederpreis: 39,80 €

www.elektor.de/serielle-schnittstellen

Das ESP8266-Praxisbuch

Der ESP8266 ist ein programmierbares WLAN-Funkmodul mit zahlreichen Schnittstellen wie UART, I²C und SPI. Das Board ist sehr preiswert und bereits für unter 3 Euro verfügbar. Die UART-Schnittstelle sorgt dabei für eine einfache Integration in Mikrocontrollerprojekte. Das ESP8266-Modul kann hervorragend mit dem Arduino zusammenarbeiten und ermöglicht ihm über die serielle Schnittstelle den Zugang zum Netzwerk und Internet.

Mitgliederpreis: 39,80 €

www.elektor.de/esp8266-buch

StromPi 2

Die StromPi 2-Erweiterungsplatine ist die Neuentwicklung der beliebten Power-Solution StromPi 1. Nicht nur, dass der Spannungsbereich auf 6-61 V erhöht wurde, um z. B. größere/mehrere Batterien oder Spannungsversorgungen anzuschließen, so liefert StromPi 2 nun auch bis zu 3 A Strom, um auch größere Projekte mit genügend Strom beliefern zu können.

Mitgliederpreis: 26,96 €

www.elektor.de/strompi-2

GSM/GPRS Projects

NEU!

Dieses (englischsprachige) Buch richtet sich an alle, die gerne Mobilfunk bzw. GSM/GPRS-Modems in Mikrocontroller-Projekte einbinden wollen. Dabei werden zwei populäre Mikrocontroller-Familien berücksichtigt: die PICs von Microchip und mit der Arduino-Plattform auch die ARM-Chips. Als PIC-Beispiel dient die verbreitete Mittelklasse-MCU PIC18F87J50 zusammen mit einem GSM-Click-Board. Bei Arduino-Uno-Projekten wird der GSM/GPRS-Shield SIM900 eingesetzt. Im Buch werden Beispiele auf der Basis beider Plattformen beschrieben.

Nach der Lektüre dieses Buchs sind Sie in der Lage, Geräte durch das Versenden von SMS-Messages zum Mikrocontroller via Handy fernzusteuern. Außerdem können Sie die Umgebungstemperatur durch einen Mikrocontroller erfassen und die Messwerte per SMS-Messages mit Hilfe von GPRS-Befehlen über das Internet an eine Cloud mit UDP- und TCP-Protokollen schicken lassen.

Mitgliederpreis: 24,95 € (frei Haus)
www.elektor.de/gsm-gprs-projects

Digitale Systeme mit FPGAs entwickeln

FPGAs sind Standard-ICs, die dem Anwender die Konfiguration von Hardwarestrukturen ermöglichen. Mit FPGAs ist die Umsetzung digitaler Systeme ab Stückzahl eins machbar. Durch die Verfügbarkeit kostenloser Lizenzen und preiswerter Entwicklungsboards ist die finanzielle Einstiegsschwelle in diese Technik niedrig. Bei der Überwindung der fachlichen Schwelle hilft dieser 6-teilige Kurs.

Mitgliederpreis: 49,00 €

www.elektor.de/digitale-systeme-fpga

AVR-Programmierung für Quereinsteiger

Atmel AVR ist eine 8-bit-Mikrocontroller-Familie des Herstellers Atmel. Diese Controller sind wegen ihres einfachen Aufbaus, ihrer leichten Programmierbarkeit, den kostenlosen Entwicklungswerkzeugen und der Verfügbarkeit in DIP-Gehäuseformen auch bei Makern äußerst beliebt. Darüber hinaus sind diese Controller bereits ab zwei Euro erhältlich. Im Arduino Uno-Board wird der ATmega328 verwendet.

Mitgliederpreis: 34,80 €

www.elektor.de/avr-prog-fuer-quereinsteiger

Arduino-Sanduhr

Auf der Basis eines Arduino Uno dient dieses coole Gadget zur Anzeige der aktuellen Uhrzeit, indem diese in eine Sandschicht „graviert" wird. Nach einer einstellbaren Zeit „löschen" zwei Vibrationsmotoren die Anzeige, bevor die neue Zeit geschrieben wird. Die Aktivitäten der Sanduhr werden durch zwei Servo-Motoren vorgenommen, die einen Pantograf-Mechanismus antreiben.

Mitgliederpreis: 107,96 €

www.elektor.de/sanduhr

Hexadoku

Sudoku für Elektroniker

Wie in jeder Ausgabe finden Sie auch in diesem Heft unser ganz spezielles Sudoku. PC, Oszilloskop und Lötkolben können sich erholen, während Ihre kleinen grauen Zellen auf Hochtouren arbeiten. Wenn Sie alle Hex-Ziffern in den grauen Kästchen herausgefunden haben, sollten Sie uns diese gleich zumailen – denn hier warten fünf Elektor-Gutscheine!

Die Regeln dieses Rätsels sind ganz einfach zu verstehen: Bei einem Hexadoku werden die Hexadezimalzahlen 0 bis F verwendet, was für Elektroniker und Programmierer ja durchaus passend ist. Füllen Sie das Diagramm mit seinen 16 x 16 Kästchen so aus, dass alle Hexadezimalzahlen von 0 bis F (also 0 bis 9 und A bis F) in jeder Reihe, jeder Spalte und in jedem Fach mit 4 x 4 Kästchen (markiert durch die dickeren schwarzen Linien) **genau einmal** vorkommen. Einige Zahlen sind bereits eingetragen, was die Ausgangssituation des Rätsels bestimmt. Wer das Rätsel löst – sprich die Zahlen in den grauen Kästchen herausfindet – kann einen von fünf Gutscheinen im Wert von 50 Euro gewinnen!

Einsenden

Schicken Sie die Lösung (die Zahlen in den grauen Kästchen) per E-Mail, Fax oder Post an:

Elektor Redaktion
Kackertstr. 10
52072 Aachen

Fax: 0241 / 955 09-013
E-Mail: hexadoku@elektor.de

Als Betreff bitte nur die Ziffern der Lösung angeben!

Einsendeschluss ist der 23. Mai 2017.

Die Gewinner des Hexadokus aus der März/April-Ausgabe stehen fest!

Die richtige Lösung ist: **314AB.**

Einen Elektor-Wertgutschein über je 50 € haben gewonnen: Sake van der Schaaf, Eugene Stemple, Michael Düren, Denis Moucharte und József Nagy.

Herzlichen Glückwunsch!

2	4		C	F		B	3	6	A						
7	B					4	6		9			A			
5		6			A		1		C		2	4	8		
			A		8	9	E	7	D				3		
	7	5			D			3				B	4		2
		2				A			F		8				E
	F		D	E	3				B		1	5		0	
		3		1	0		5			9					
		7		B	E		8			C					
	5		8	9	1				2		6	C		3	
		C				F			3		4				8
	6	E			C			F				9	B		0
			5		4	1	C	0	6				2		
C		F			7		0		4		3	D	9		
6	3					8	D		1			0			
A	D		2	5		6	B	9	8						

B	9	8	0	5	6	4	A	D	C	E	1	F	3	7	2
5	A	7	D	1	F	C	8	0	2	3	6	E	9	4	B
C	E	1	2	B	9	7	3	F	4	8	A	D	5	6	0
F	4	3	6	D	E	0	2	9	5	7	B	C	1	A	8
1	7	B	C	9	A	2	D	5	3	4	F	0	E	8	6
E	D	5	A	F	0	8	7	1	6	C	9	2	4	B	3
2	F	4	3	C	5	B	6	7	8	0	E	9	A	1	D
0	6	9	8	E	3	1	4	A	B	D	2	5	C	F	7
3	0	C	F	2	7	E	B	6	9	A	D	4	8	5	1
A	5	D	4	8	1	9	0	3	E	F	7	6	B	2	C
6	8	E	1	3	C	D	5	B	0	2	4	A	7	9	F
7	B	2	9	6	4	A	F	8	1	5	C	3	D	0	E
4	C	F	B	7	D	3	1	2	A	9	0	8	6	E	5
8	2	A	7	0	B	5	E	4	D	6	3	1	F	C	9
9	3	6	5	A	2	F	C	E	7	1	8	B	0	D	4
D	1	0	E	4	8	6	9	C	F	B	5	7	2	3	A

Der Rechtsweg ist ausgeschlossen. Mitarbeiter der in der Unternehmensgruppe Elektor International Media B.V. zusammengeschlossenen Verlage und deren Angehörige sind von der Teilnahme ausgeschlossen.

Mit Business Plus erhalten Sie ein Rundum-Sorglospaket für Ihr Business.

Armin Tischler, Tekkie bei Conrad seit 1995

Business Plus

Ihre Vorteile auf einen Blick:

Die Tek Flat - 1 Jahr keine Versandkosten
Kostenlose Rücksendung innerhalb 30 Tagen
24h Lieferung - schnell zum Produkt
36 Monate Langzeit-Garantie - Businesskarte
Vorteile übertragbar - auf bis zu 5 Besteller
Ihre persönliche Hotline - besser beraten
Newsletter - schneller informiert
Innovation News - kostenloses Fachmagazin
Verlängertes Zahlungsziel - 30 Tage
Abholservice - in der Filiale

Business Plus
33,57 €*
zzgl. MwSt.

Jetzt informieren unter:
conrad.biz/businessplus

CONRAD
Business Supplies

* Gültig für 1 Jahr ab Kauf, für Bestellungen innerhalb Deutschlands. Zusätzliche Serviceaufschläge sind nicht inbegriffen. Mit dem Kauf von Business Plus können bis zu 5 Ansprechpartner unter einer Kundennummer aus Ihrem Unternehmen bestelllen. Irrtümer und Druckfehler vorbehalten. Die AGBs finden Sie unter conrad.de.

reichelt elektronik

Technik verbindet.

BESTELLT | BELIEFERT | BEGEISTERT

- ✓ Mehr als 75.000 Produkte
- ✓ Top-Preis-Leistungsverhältnis
- ✓ Hohe Verfügbarkeit und 24-h-Lieferservice
- ✓ Starke Marken und echte Qualität

reichelt.de | @reichelt_el | google.com/+reichelt

Foto: Foto- und Bilderwerk

ENTDECKEN SIE FLUKE-MESSTECHNIK ZUM SONDERPREIS!

FLUKE

Präzise, benutzerfreundlich, sicher und zuverlässig

TRMS Digital-Multimeter

- Echteffektivmessung von Spannung und Strom, 1000 V / 10 A
- 0,1 % Grundgenauigkeit
- LCD, 6.000 Counts Auflösung
- manuelle und automatische Bereichswahl
- Frequenz-, Kapazitäts- und Widerstandsmessung

SET INKL. TASCHE & MESSLEITUNGEN

EN 61010-1 CAT III 1000 V
EN 61010-1 CAT IV 600 V

statt 337,95 FLUKE 175 PROMO

44% SPAREN 189,00

Für präzise Messungen von Laststrom, Spannung oder Widerstand

Echteffektiv-Stromzange

- Wechselstrommessung bis zu 400 A
- Gleich- und Wechselspannungsmessung bis 600 V
- TRMS-Messung für präzise Messresultate bei nichtlinearen Signalen
- Widerstandsmessung bis zu 40 kOhm
- Temperatur-, Kapazitäts- und Frequenzmessung

SET INKL. HOLSTER & KOFFER

EN 61010-1 CAT IV 300 V
EN 61010-1 CAT III 600 V

statt 370,10 FLUKE 325 PROMO

38% SPAREN 229,00

FLUKE

Stellt die Sicherheit und Funktionalität Ihrer Geräte sicher!

Gerätetester-Kit für Messungen nach VDE 0701/0702

- inkl. Koffer, Software u.v.m.

EN 61010-1 CAT II 300 V

statt 2684,00 FLUKE 6500-2 KIT

32% SPAREN 1829,00

FLUKE

Über 97% zufriedene Kunden!

★★★★☆

www.shopauskunft.de

SHOPAUSKUNFT.DE EMPFEHLUNG

Tagespreise · Preisstand: 21. 3. 2017, Preise in € inkl. gesetzl. MwSt., zzgl. Versandkosten, reichelt elektronik, Elektronikring 1, 26452 Sande (D)

JETZT NEWSLETTER ABONNIEREN & GEWINNEN!

Wir verlosen monatlich unter allen neuen Newsletter-Abonnenten ein technisches Highlight!

JETZT MITMACHEN ► http://rch.lt/v3

www.reichelt.de

BESTELLHOTLINE: +49 (0)4422 955-333